Handbücher/Manuals

art of automation

VIPA
Gesellschaft für Visualisierung
und Prozessautomatisierung mbH

Ohmstraße 4
D-91074 Herzogenaurach
Tel.: +49-9132-744-0
Fax: +49-9132-744-144
Internet: www.vipa.de
E-Mail: Info@vipa.de

# Handbuch

## VIPA CPU 24x

Best.-Nr.: VIPA HB99D_CPU
Rev. 08/32

Die Angaben in diesem Handbuch erfolgen ohne Gewähr. Änderungen des Inhalts können jederzeit ohne Vorankündigung erfolgen.

© Copyright 2008 VIPA, Gesellschaft für Visualisierung und Prozessautomatisierung mbH
Ohmstraße 4, D-91074 Herzogenaurach,
Tel.: +49 (91 32) 744 -0
Fax.: +49 (91 32) 744-144
EMail: info@vipa.de
http://www.vipa.de

**Hotline: +49 (91 32) 744-114**

Alle Rechte vorbehalten

**Haftungsausschluss**

Der Inhalt dieses Handbuchs wurde auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft.

Dennoch können Abweichungen nicht ausgeschlossen werden. Die Angaben in diesem Handbuch werden regelmäßig überprüft und erforderliche Korrekturen sind in den nachfolgenden Auflagen enthalten.

Für Verbesserungsvorschläge sind wir dankbar.

**Warenzeichen**

| | |
|---|---|
| VIPA® | ist eingetragenes Warenzeichen der VIPA Gesellschaft für Visualisierung und Prozessautomatisierung mbH |
| SIMATIC® | ist eingetragenes Warenzeichen der Siemens AG. |
| STEP® 5 | ist eingetragenes Warenzeichen der Siemens AG. |

Alle ansonsten im Text genannten Warenzeichen sind Warenzeichen der jeweiligen Inhaber und werden als geschützt anerkannt.

# Über dieses Handbuch

Dieses Handbuch beschreibt den Einsatz der CPU 24x im System 200V von VIPA.

Sie erfahren näheres über die Hardware, die Programmierung sowie über Integrierte Funktionen.

**Überblick**

**Teil 1: Grundlagen**

Im Rahmen dieser Einleitung folgen Hinweise im Umgang und Informationen über Einsatzbereiche und Anwendung der CPU-Baugruppen.

Sie erfahren etwas über die Arbeitsweise der CPU 24x.

**Teil 2: Hardwarebeschreibung**

Die CPU erhalten Sie in verschiedenen Ausführungen (CPU 24x, CPU 24x DP, CPU 24x NET) auf die in diesem Kapitel weiter eingegangen werden soll. Neben der Hardwarebeschreibung finden Sie hier auch Anweisungen zur Inbetriebnahme und zum Einsatz der Speichermodule.

Mit einer Übersicht der integrierten FBs und OBs und den Technischen Daten wird das Kapitel abgeschlossen.

**Teil 3: Einsatz CPU 24x mit E/A Modulen**

In diesem Kapitel ist der Einsatz der CPU 24x, CPU 24x DP, CPU 24x NET mit den Peripherie-Modulen des System 200V beschrieben.

**Teil 4: Einsatz CPU 24x2BT10 NET**

Einsatz, Projektierung, Funktionsbeschreibung, Programmierung der CPU 24x2BT10 NET.

**Teil 5: Einsatz CPU 24x2BT01 NET**

Einsatz, Projektierung, Funktionsbeschreibung, Programmierung der CPU 24x2BT01 NET.

**Teil 6: Einsatz CPU 24x DP**

Einsatz, Projektierung, Beschreibung und Parametrierung der CPU 24x DP unter Profibus.

**Teil 7: Betriebszustände**

Es werden die Betriebszustände Stop, Run, und Anlauf beschrieben.

Außerdem wird die Zyklische -, Zeitgesteuerte -, und die Alarmgesteuerte Programmbearbeitung näher erläutert.

**Teil 8: Einführung in die Programmiersprache**

Dieses Kapitel beschreibt das Programmieren von Automatisierungsaufgaben. Es wird erklärt, wie man Programme erstellt und welche Bausteine zur Gliederung eines Programms eingesetzt werden können. Außerdem finden Sie eine Übersicht der verschiedenen Zahlendarstellungsarten, die die Programmiersprache kennt.

**Teil 9: Operationen**

Der Befehlsvorrat wird näher beschrieben. Grundoperationen, Ergänzende Operationen und Systemoperationen werden anhand von Beispielen näher erläutert.

**Teil 10: Integrierte Bausteine**

Die Integrierten FBs, die Organisationsbausteine und der Default-DB 1 (für die Parametrierung interner Funktionen) werden in diesem Kapitel beschrieben.

**Teil 11: Uhrfunktionen**

In diesem Kapitel ist der Aufbau und die Parametrierung der integrierten Uhr beschrieben. Zur Zeit besitzen die CPU-Typen CPU 243 und CPU 244 eine integrierte Uhr.

**Teil 12: Alarm- und Zeitgesteuerte Bearbeitung**

Sie erfahren etwas über die Programmbearbeitung, bzw. die Unterbrechung des zyklischen Programmbetriebs mit der CPU 24x.

# Inhaltsverzeichnis

**Benutzerhinweise** ..... **1**
**Sicherheitshinweise** ..... **2**
**Teil 1 Grundlagen** ..... **1-1**
Sicherheitshinweis für den Benutzer ..... 1-2
Allgemeines ..... 1-3
Einsatzbereiche ..... 1-4
Leistungsmerkmale ..... 1-5
Gerätevarianten ..... 1-6
Arbeitsweise einer CPU ..... 1-7
Programme der CPU 24x ..... 1-8
Operanden der CPU 24x ..... 1-8
**Teil 2 Hardwarebeschreibung** ..... **2-1**
Systemübersicht ..... 2-2
Aufbau ..... 2-6
Komponenten ..... 2-8
Blockschaltbild ..... 2-16
Inbetriebnahme ..... 2-17
Anlaufverhalten ..... 2-18
URLÖSCHEN und Neustart ..... 2-19
Anwenderprogramm laden und sichern ..... 2-20
Testfunktion STATUS/STATUS VAR und STEUERN ..... 2-22
MMC-Speicher-Modul ..... 2-24
Speicherbereiche ..... 2-25
USTACK - Ausgabe der Systemdaten mit dem PG ..... 2-27
Bedeutung der USTACK-Anzeigen ..... 2-32
Integrierte FBs und OBs ..... 2-34
Technische Daten ..... 2-36
**Teil 3 Einsatz CPU 24x mit E/A Modulen** ..... **3-1**
Einsatz in Verbindung mit E/A-Modulen ..... 3-2
Automatische Vergabe von Peripherieadressen ..... 3-3
Manuelle Vergabe von Peripherieadressen ..... 3-5
Parametrierung von Peripheriemodulen ..... 3-6
**Teil 4 Einsatz CPU 24x-2BT10** ..... **4-1**
Industrial Ethernet in der Automatisierung ..... 4-2
ISO/OSI-Schichtenmodell ..... 4-3
Grundbegriffe ..... 4-6
Protokolle ..... 4-7
IP-Adresse und Subnetz ..... 4-10
Planung eines Netzwerks ..... 4-12
Kommunikationsmöglichkeiten des CP ..... 4-14
Kommunikation programmieren ..... 4-16
SEND/RECEIVE im SPS-Anwenderprogramm ..... 4-23
Kopplung mit Fremdsystemen ..... 4-27
NCM-Diagnose - Hilfe zur Fehlersuche ..... 4-30

**Teil 5 Einsatz CPU 24x-2BT01 ..... 5-1**
Grundlagen ..... 5-2
Planung eines Netzwerks ..... 5-7
Standardisierung und Normen ..... 5-9
Ethernet- und IP-Adresse ..... 5-10
Projektierung der CPU 24x NET ..... 5-12
Beispiel zur Projektierung ..... 5-16
Anlaufverhalten ..... 5-29
Systemeigenschaften der CPU 24x NET ..... 5-30
Kopplung mit Fremdsystemen ..... 5-32
Status- und Fehleranzeigen ..... 5-36
Testprogramm für TCP/IP-Verbindungen ..... 5-44
**Teil 6 Einsatz CPU 24x DP ..... 6-1**
Grundlagen ..... 6-2
Projektierung der CPU 24x DP ..... 6-6
Default-Adressierung mit DB 1 ändern ..... 6-9
Zugriff auf Parameterdaten ..... 6-12
Parametrierung der direkt gesteckten E/A-Module ..... 6-13
Diagnosefunktionen der CPU 24x DP ..... 6-14
Statusmeldung intern an CPU ..... 6-17
Aufbaurichtlinien ..... 6-19
Inbetriebnahme ..... 6-24
Einsatz der Diagnose-LEDs ..... 6-27
Beispiel ..... 6-28
**Teil 7 Betriebszustände ..... 7-1**
Einführung und Übersicht ..... 7-2
Programmbearbeitungsebenen ..... 7-3
Betriebszustand STOP ..... 7-6
Betriebszustand ANLAUF ..... 7-8
Betriebszustand RUN ..... 7-11
Programmablauf ..... 7-12
**Teil 8 Einführung in die Programmiersprache ..... 8-1**
Vorgehensweise bei der Programmierung ..... 8-2
Erstellung eines Programms ..... 8-4
Programmstruktur ..... 8-8
Bausteinarten ..... 8-9
Programmbearbeitung ..... 8-22
Bearbeiten von Bausteinen ..... 8-32
Zahlendarstellung ..... 8-34
Fehlerdiagnose im Programm ..... 8-35
**Teil 9 Operationen ..... 9-1**
Einleitung ..... 9-2
Grundoperationen ..... 9-3
Ergänzende Operationen ..... 9-40
Systemoperationen ..... 9-66
Anzeigenbildung ..... 9-76
Programmbeispiele ..... 9-79

**Teil 10 Integrierte Bausteine ..... 10-1**
Integrierte Funktionen ..... 10-2
FB 238 - COMPR (komprimieren) ..... 10-3
FB 239 - DELETE (löschen) ..... 10-4
FB 240 - COD:B4 (Codewandlung BCD/DUAL) ..... 10-5
FB 241 - COD:16 (Codewandler DUAL/BCD) ..... 10-6
FB 242 - MUL:16 (Multiplizierer) ..... 10-7
FB 243 - DIV:16 (Dividierer) ..... 10-8
FB 250 - RLG:AE (Analogwert lesen und normieren) ..... 10-9
FB 251 - RLG:AA (Analogwert ausgeben) ..... 10-12
Beispiel für eine Analogwertverarbeitung ..... 10-14
Integrierte Hantierungsbausteine ..... 10-17
Die Parameter der Hantierungsbausteine ..... 10-18
Parameterbeschreibung der Hantierungsbausteine ..... 10-19
Parametrierung von SSNR, A-NR, ANZW und BLGR ..... 10-21
Indirekte Parametrierung von Quell- und Zielangaben ..... 10-24
Tabelle über die möglichen QTYP/ZTYP - Parameter ..... 10-25
Aufbau des Anzeigenwortes ..... 10-27
Status und Fehleranzeige im Anzeigenwort ..... 10-28
Anzeigenwort ..... 10-29
Längen - Wort ..... 10-35
Aufbau des Parametrierfehler - Anzeigenbytes ..... 10-36
Einstellbare Blockgrößen ..... 10-37
FB 244 - SEND ..... 10-38
FB 245 - RECEIVE ..... 10-41
FB 246 - FETCH ..... 10-44
FB 247 - CONTROL ..... 10-45
FB 248 - RESET ..... 10-46
FB 249 - SYCHRON ..... 10-47
Integrierte Sonder-Organisationsbausteine ..... 10-48
OB 31 - Zykluszeittriggerung ..... 10-49
OB 160 - Variable Zeitschleife ..... 10-50
OB 251 - PID-Regelalgorithmus ..... 10-51
Integrierter Baustein DB 1 ..... 10-61
**Teil 11 Uhrfunktionen ..... 11-1**
Allgemeines ..... 11-2
Parametrierung der integrierten Uhr ..... 11-3
Aufbau des Uhrdatenbereichs ..... 11-7
Aufbau des Statusworts ..... 11-10
Uhr stellen und lesen ..... 11-12
Weckfunktionen ..... 11-19
Betriebsstundenzähler ..... 11-22
**Teil 12 Alarm- und Zeitgesteuerte Bearbeitung ..... 12-1**
Programmierung von Alarmbausteinen ..... 12-2
Zeitgesteuerte Programmbearbeitung ..... 12-4
Zeitdauergesteuerte Programmbearbeitung ..... 12-6
Priorität und Reaktionszeit ..... 12-7
Alarm-Diagnosedaten ..... 12-11

**Anhang ..... A-1**
Befehlsliste ..... A-1
Maschinencode ..... B-1
Index ..... C-1

# Benutzerhinweise

**Zielsetzung und Inhalt**

Dieses Handbuch beschreibt die CPU 24x mit allen Produktvarianten. Beschrieben wird Aufbau, Projektierung und Anwendung.

Zusammen mit der CPU 24x lassen sich alle System-200V-Komponenten der VIPA einsetzen.

**Zielgruppe**

Das Handbuch ist geschrieben für Anwender mit Grundkenntnissen in der Automatisierungstechnik und in der SPS-Programmierung.

**Aufbau des Handbuchs**

Das Handbuch ist zur Zeit gegliedert in Kapitel. Jedes Kapitel beschreibt eine abgeschlossene Thematik.

**Orientierung im Dokument**

Als Orientierungshilfe stehen im Handbuch zur Verfügung:

- Gesamt-Inhaltsverzeichnis am Anfang des Handbuchs
- Übersicht der beschriebenen Themen am Anfang jedes Kapitels
- Stichwortverzeichnis (Index) am Ende des Handbuchs

**Verfügbarkeit**

Das Handbuch ist verfügbar in:

- gedruckter Form auf Papier
- in elektronischer Form als PDF-Datei (Adobe Acrobat Reader)

**Piktogramme Signalwörter**

Besonders wichtige Textteile sind mit folgenden Piktogrammen und Signalworten ausgezeichnet:

**Gefahr!**

Unmittelbar drohende oder mögliche Gefahr.

Personenschäden sind möglich.

**Achtung!**

Bei Nichtbefolgen sind Sachschäden möglich.

**Hinweis!**

Zusätzliche Informationen und nützliche Tipps

# Sicherheitshinweise

**Bestimmungs-gemäße Verwendung**

Die CPU 24x ist konstruiert und gefertigt für

- alle VIPA System-200V-Komponenten
- Kommunikation und Prozesskontrolle
- Allgemeine Steuerungs- und Automatisierungsaufgaben
- den industriellen Einsatz
- den Betrieb innerhalb der in den technischen Daten spezifizierten Umgebungsbedingungen
- den Einbau in einen Schaltschrank

**Gefahr!**

Das Gerät ist nicht zugelassen für den Einsatz

- in explosionsgefährdeten Umgebungen (EX-Zone)

**Dokumentation**

Handbuch zugänglich machen für alle Mitarbeiter in

- Projektierung
- Installation
- Inbetriebnahme
- Betrieb

**Vor Inbetriebnahme und Betrieb der in diesem Handbuch beschriebenen Komponenten unbedingt beachten:**

- Änderung am Automatisierungssystem nur im spannungslosen Zustand vornehmen!
- Anschluss und Änderung nur durch ausgebildetes Elektro-Fachpersonal
- Nationale Vorschriften und Richtlinien im jeweiligen Verwenderland beachten und einhalten (Installation, Schutzmaßnahmen, EMV ...)

**Entsorgung**

**Zur Entsorgung des Geräts nationale Vorschriften beachten!**

# Teil 1 Grundlagen

**Überblick**

Im Rahmen dieser Einleitung folgen Hinweise im Umgang und Informationen über Einsatzbereiche und Anwendung der CPU-Baugruppen.

Sie erhalten ferner einige Vorschläge, wie Sie beim Programmieren vorgehen können und erfahren, welche für die Programmierung wichtige Kenndaten die CPU hat.

**Inhalt**


| Thema | Seite |
|---|---|
| **Teil 1 Grundlagen** | **1-1** |
| Sicherheitshinweis für den Benutzer | 1-2 |
| Allgemeines | 1-3 |
| Einsatzbereiche | 1-4 |
| Leistungsmerkmale | 1-5 |
| Gerätevarianten | 1-6 |
| Arbeitsweise einer CPU | 1-7 |
| Programme der CPU 24x | 1-8 |
| Operanden der CPU 24x | 1-8 |

# Sicherheitshinweis für den Benutzer

**Handhabung elektrostatisch gefährdeter Baugruppen**

VIPA-Baugruppen sind mit hochintegrierten Bauelementen in MOS-Technik bestückt. Diese Bauelemente sind hoch empfindlich gegenüber Überspannungen, die z.B. bei elektrostatischer Entladung entstehen.

Zur Kennzeichnung dieser gefährdeten Baugruppen wird nachfolgendes Symbol verwendet:

Das Symbol befindet sich auf Baugruppen, Baugruppenträgern oder auf Verpackungen und weist so auf elektrostatisch gefährdete Baugruppen hin.

Elektrostatisch gefährdete Baugruppen können durch Energien und Spannungen zerstört werden, die weit unterhalb der Wahrnehmungsgrenze des Menschen liegen. Hantiert eine Person, die nicht elektrisch entladen ist, mit elektrostatisch gefährdeten Baugruppen, können diese Spannungen auftreten und zur Beschädigung von Bauelementen führen und so die Funktionsweise der Baugruppen beeinträchtigen oder die Baugruppe unbrauchbar machen. Auf diese Weise beschädigte Baugruppen werden in den wenigsten Fällen sofort als fehlerhaft erkannt. Der Fehler kann sich erst nach längerem Betrieb einstellen.

Durch statische Entladung beschädigte Bauelemente können bei Temperaturänderungen, Erschütterungen oder Lastwechseln zeitweilige Fehler zeigen.

Nur durch konsequente Anwendung von Schutzeinrichtungen und verantwortungsbewusste Beachtung der Handhabungsregeln lassen sich Funktionsstörungen und Ausfälle an elektrostatisch gefährdeten Baugruppen wirksam vermeiden.

**Versenden von Baugruppen**

Verwenden Sie für den Versand immer die Originalverpackung.

**Messen und Ändern von elektrostatisch gefährdeten Baugruppen**

Bei Messungen an elektrostatisch gefährdeten Baugruppen sind folgende Dinge zu beachten:

- Potentialfreie Messgeräte sind kurzzeitig zu entladen.
- Verwendete Messgeräte sind zu erden.

Bei Änderungen an elektrostatisch gefährdeten Baugruppen ist darauf zu achten, dass ein geerdeter Lötkolben verwendet wird.

**Achtung!**

Bei Arbeiten mit und an elektrostatisch gefährdeten Baugruppen ist auf ausreichende Erdung des Menschen und der Arbeitsmittel zu achten.

# Allgemeines

**Befehls-kompatibilität**

Die CPU 24x ist befehlskompatibel zur Programmiersprache STEP® 5 von Siemens und hat FBs und OBs integriert.

**Parametrierung über DB1**

Die Parametrierung der System-200V-Module erfolgt über den DB1 unter Zuhilfenahme der Parametriersoftware WinNCS von VIPA.

**Integriertes Netzteil**

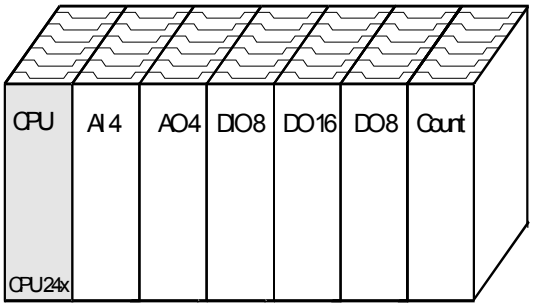
Die CPU 24x hat ein Netzteil integriert, das über die Front mit DC24V zu versorgen ist. Das Netzteil ist gegen Verpolung und Überstrom geschützt.

# Einsatzbereiche

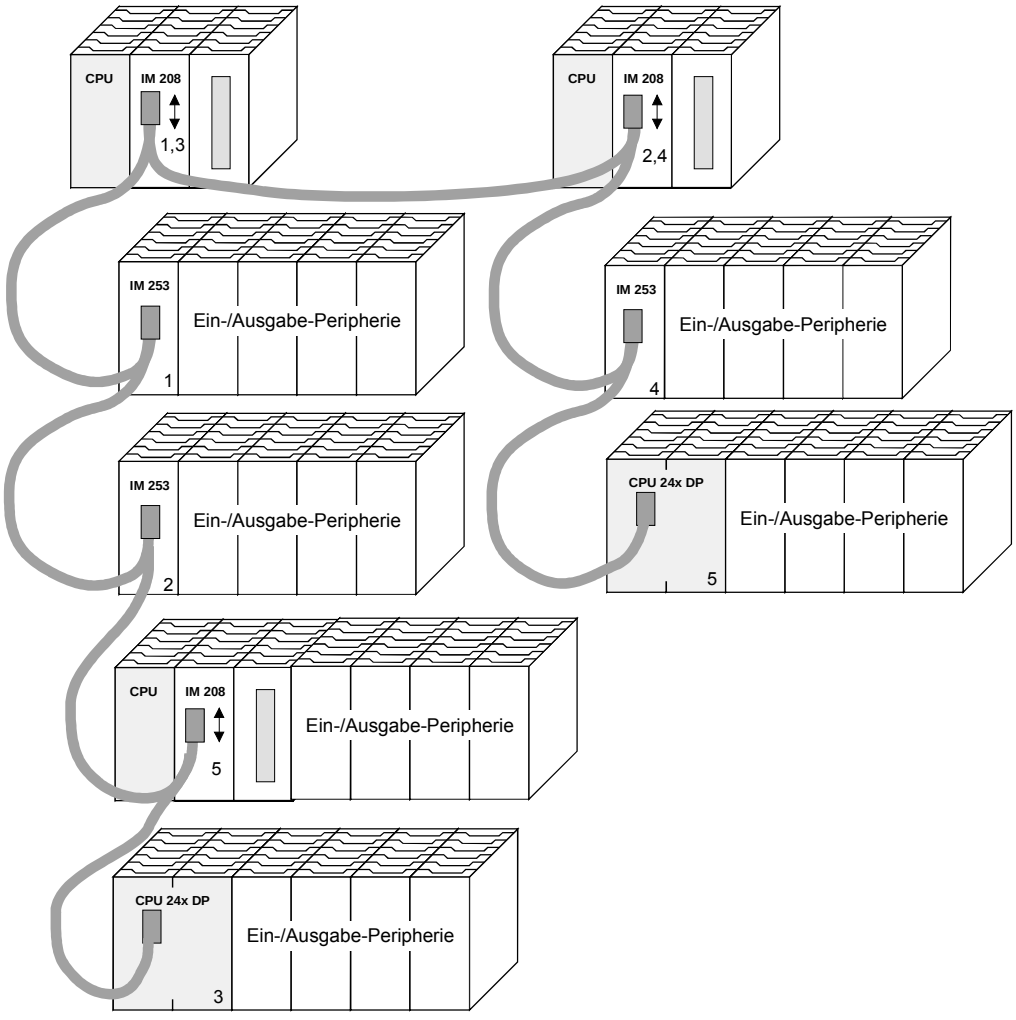
Mit dieser CPU-Serie haben Sie Zugriff auf die Peripherie-Module des System 200V von VIPA. Sie können über standardisierte Befehle und Programme Sensoren abfragen und Aktoren steuern. Eine CPU kann maximal 32 Module ansprechen. Über die normierte PG/OP-Schnittstelle haben Sie Zugriff auf andere Geräte.

**Anwendungs-beispiel**

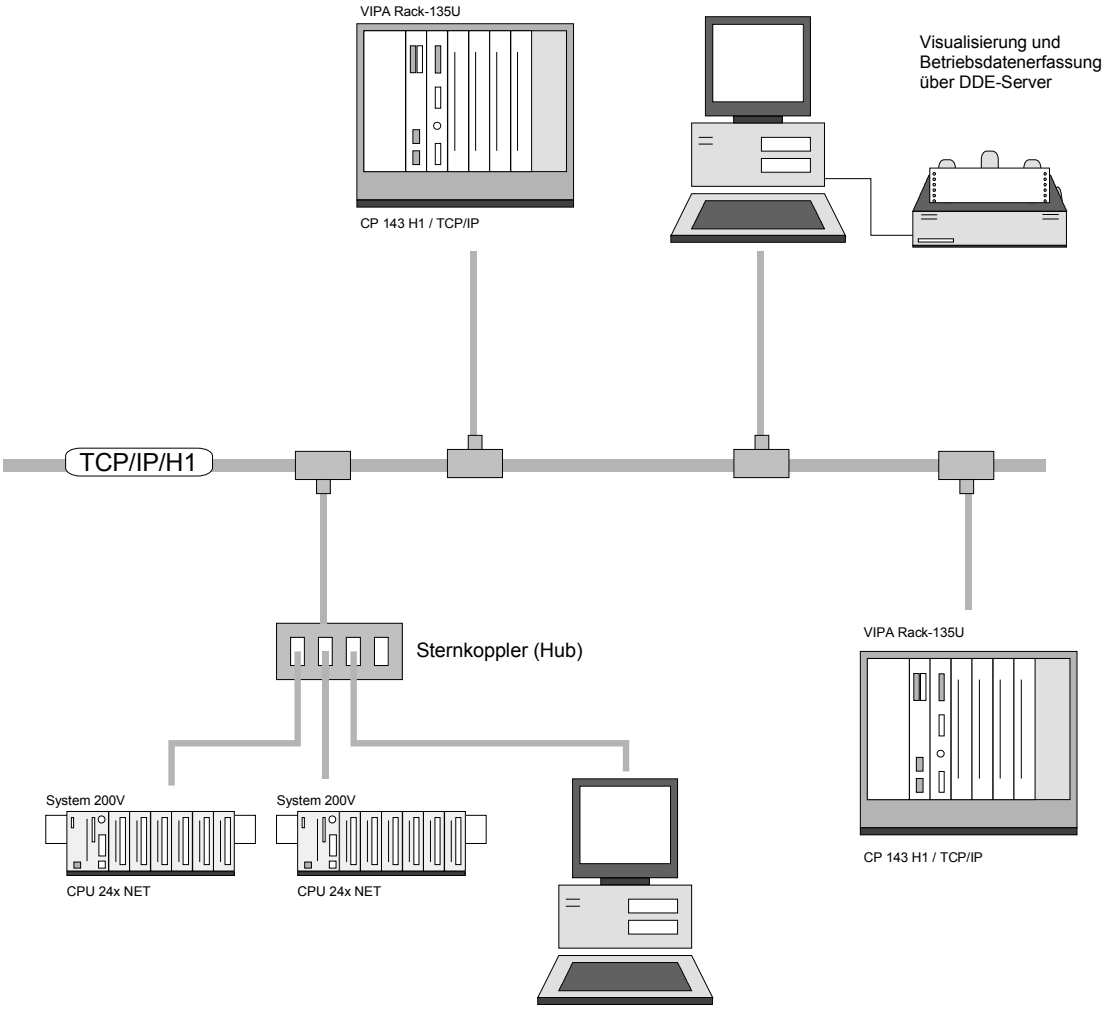
*Zentrale Kompakt-Lösung*




*Dezentraler Einsatz unter Profibus*



*Einsatz unter TCP/P bzw. H1*




# Leistungsmerkmale

Die SPS-CPUs dienen als CPUs zur Programmverarbeitung.
Sie bedient die Ein-/Ausgabe-Module und verarbeitet Daten der Funktionsmodule.

- Schnelle Programmierung durch Kompatibilität zu STEP® 5 von Siemens.
- Platzeinsparung durch kompakten Aufbau.
- Hohe Flexibilität durch Anschluss von bis zu 32 Funktionsmodulen (DIG E/A, ANA E/A, SSI, Impulszähler, Kommunikation etc.).
- Schnelle Nachrüstung von Funktionsmodulen durch steckbare Busverlängerung.
- Anwenderfreundliche Wartung mittels PG.
- Einfacher Anschluss verschiedener Operator Panels über PG/AG Schnittstelle AS 511.
- Optional mit Ethernet- oder Profibus-Schnittstelle.

# Gerätevarianten

Die CPU 24x ist bei VIPA in 3 Ausführungen erhältlich, die sich auch äußerlich unterscheiden:

**Ausführungen**

- **CPU 24x** SPS-CPU
- **CPU 24x NET** SPS-CPU mit Ethernet-Anschaltung
- **CPU 24x DP** SPS-CPU mit Profibus-Slave

Alle CPU 24x sind in 4 CPU-Leistungsklassen lieferbar als 241, 242, 243 und 244. Innerhalb dieser 4 Leistungsklassen gibt es optisch keinen Unterschied. Mit steigender Nummer steigt auch der Leistungsumfang einer CPU 24x. Nachfolgend sind die Leistungsklassen aufgeführt.

**Leistungsklassen**

| Daten | CPU 241 | CPU 242 | CPU 243 | CPU 244 |
|---|---|---|---|---|
| Speicher | 8KB | 32KB | 52KB | 104KB |
| Batteriepuffer / Uhr | ja / nein | ja / nein | ja / ja | ja / ja |
| Merker / S-Merker | 1024 / 0 | 2048 / 0 | 2048 / 0 | 2048 / 8192 |
| Zeiten / Zähler | 32 / 32 | 64 / 64 | 128 / 128 | 128 / 128 |
| Adresse E/A digital | 448 | 512 | 1024 | 1024 |
| Adresse E/A analog | 16 | 32 | 256 | 256 |
| Bearbeitungszeit | 1,8ms | 1,8ms | 1,8ms | 1,8ms |
| Programmbearbeitung | zykl. | zykl., Zeit | zykl., Zeit | zykl., Zeit, Alarm |
| Bestelldaten | | | | |
| CPU 24x | 241-1BA01 | 242-1BA01 | 243-1BA01 | 244-1BA01 |
| CPU 24x NET | 241-2BT01 | 242-2BT01 | 243-2BT01 | 244-2BT01 |
| CPU 24x NET | 241-2BT10 | 242-2BT10 | 243-2BT10 | 244-2BT10 |
| CPU 24x DP | 241-2BP01 | 242-2BP01 | 243-2BP01 | 244-2BP01 |

**Hinweis!**
Soweit nichts anderes erwähnt ist, beziehen sich die in diesem Handbuch gemachten Angaben auf alle CPUs der CPU 24x-Familie von VIPA!

# Arbeitsweise einer CPU

**Allgemein**

Diese CPUs sind für kleine und mittlere Anwendungen mit integriertem 24V-Netzteil. Die CPU enthält einen Standardprozessor mit internem Programmspeicher. In Verbindung mit der System-200V-Peripherie erhalten Sie ein leistungsfähiges Gerät zur Prozessautomatisierung innerhalb der System-200V-Familie.

In einer CPU gibt es folgende Arbeitsweisen:

- zyklische Bearbeitung
- zeitgesteuerte Bearbeitung
- alarmgesteuerte Bearbeitung
- Bearbeitung nach Priorität

**zyklische Bearbeitung**

Die **zyklische** Bearbeitung stellt den Hauptanteil aller Vorgänge in der CPU. In einem endlosen Zyklus werden die gleichen Bearbeitungsfolgen wiederholt.

**zeitgesteuerte Bearbeitung**

Erfordern Prozesse in konstanten Zeitabschnitten Steuersignale, so können Sie neben dem zyklischen Ablauf **zeitgesteuert** bestimmte Aufgaben durchführen z.B. zeitunkritische Überwachungsfunktionen im Sekundenraster.

**alarmgesteuerte Bearbeitung**

Soll auf ein Prozesssignal besonders schnell reagiert werden, so ordnen Sie diesem einen **alarmgesteuerten** Bearbeitungsabschnitt zu. Ein Alarm kann in Ihrem Programm eine Bearbeitungsfolge aktivieren.

**Bearbeitung nach Priorität**

Die oben genannten Bearbeitungsarten werden von der CPU nach Wichtigkeitsgrad behandelt **(Priorität)**. Da auf ein Zeit- oder Alarmereignis schnell reagiert werden muss, unterbricht zur Bearbeitung dieser hochprioren Ereignisse die CPU die zyklische Bearbeitung, reagiert auf diese Ereignisse und setzt danach die zyklische Bearbeitung wieder fort. Die zyklische Bearbeitung hat daher die niedrigste Priorität.

# Programme der CPU 24x

Das in jeder CPU vorhandene Programm unterteilt sich in:

- Systemprogramm,
- Anwenderprogramm.

**Systemprogramm** Das Systemprogramm organisiert alle Funktionen und Abläufe der CPU, die nicht mit einer spezifischen Steuerungsaufgabe verbunden sind.

**Anwender-programm** Hier finden Sie alle Funktionen, die zur Bearbeitung einer spezifischen Steuerungsaufgabe erforderlich sind. Schnittstellen zum Systemprogramm stellen die Operationsbausteine zur Verfügung.

# Operanden der CPU 24x

Die CPU 24x stellt Ihnen für das Programmieren folgende Operandenbereiche zur Verfügung:

**Prozessabbild und Peripherie** Auf das Prozessabbild der Aus- und Eingänge PAA/PAE kann Ihr Anwenderprogramm sehr schnell zugreifen. Sie haben Zugriff auf folgende Datentypen:

- Einzelbits
- Bytes
- Wörter
- Doppelwörter.

Sie können mit Ihrem Anwenderprogramm über den Bus direkt auf Peripheriebaugruppen zugreifen. Folgende Datentypen sind möglich:

- Bytes
- Wörter.

**Merker (M- und S-Merker)**

Der Merkerbereich ist ein Speicherbereich, auf den Sie über Ihr Anwenderprogramm mit entsprechenden Operationen zugreifen können. Verwenden Sie den Merkerbereich für oft benötigte Arbeitsdaten.

Sie können auf folgende Datentypen zugreifen:

- Einzelbits
- Bytes
- Wörter
- Doppelwörter

Die CPU 24x enthält einen zusätzlichen Merkerbereich, den S-Merkerbereich. S-Merker lassen sich nicht als Aktualoperanden bei Funktionsbaustein-Aufrufen benutzen.
Voraussetzung für den Einsatz von S-Merkern ist die PG-Systemsoftware "S5-DOS" ab Version 3.0 oder "S5-DOS/MT" ab Version 1.0 von Siemens oder Sie setzen das MC5-Programmierpaket von VIPA ein.

**Zeiten und Zähler**

Sie können mit Ihrem Anwendungsprogramm eine Zeitzelle mit einem Wert zwischen 10ms und 9990s laden. Sobald Ihr Anwenderprogramm eine Startoperation ausführt, wird dieser Zeitwert um ein durch Sie vorgegebenes Zeitraster dekrementiert, bis Null erreicht wird.

Für den Einsatz von Zählern können Sie Zählerzellen mit einem Anfangswert laden (max. 999) und diesen hinauf- bzw. herunterzählen.

**Datenbausteine**

Ein Datenbaustein enthält Konstanten bzw. Variablen im Byte-, Wort- oder Doppelwortformat. Mit Operanden können Sie immer auf den aktuellen Datenbaustein zugreifen.

Sie haben Zugriff auf folgende Datentypen:

- Einzelbits
- Bytes
- Wörter
- Doppelwörter

# Teil 2 Hardwarebeschreibung

**Überblick**

Die CPUs 24x erhalten Sie in verschiedenen Ausführungen auf die in diesem Kapitel weiter eingegangen werden soll. Neben der Hardwarebeschreibung finden Sie hier auch Anweisungen zur Inbetriebnahme und zum Einsatz der Speichermodule.

Mit einer Übersicht der integrierten FBs und OBs und den Technischen Daten wird das Kapitel abgeschlossen.

**Inhalt**


| Thema | Seite |
|---|---|
| **Teil 2 Hardwarebeschreibung** | **2-1** |
| Systemübersicht | 2-2 |
| Aufbau | 2-6 |
| Komponenten | 2-8 |
| Blockschaltbild | 2-16 |
| Inbetriebnahme | 2-17 |
| Anlaufverhalten | 2-18 |
| URLÖSCHEN und Neustart | 2-19 |
| Anwenderprogramm laden und sichern | 2-20 |
| Testfunktion STATUS/STATUS VAR und STEUERN | 2-22 |
| MMC-Speicher-Modul | 2-24 |
| Speicherbereiche | 2-25 |
| USTACK - Ausgabe der Systemdaten mit dem PG | 2-27 |
| Bedeutung der USTACK-Anzeigen | 2-32 |
| Integrierte FBs und OBs | 2-34 |
| Technische Daten | 2-36 |

# Systemübersicht

Bei VIPA sind aus der CPU-24x-Familie 3 Ausführungen in je 4 Varianten erhältlich:

- **CPU 24x** SPS-CPU
- **CPU 24x NET** SPS-CPU mit CP243 Ethernet-Anschaltung
- **CPU 24x DP** SPS-CPU mit Profibus-Slave

Alle CPU 24x sind in den Varianten 241, 242, 243 und 244 erhältlich.

**CPU 24x**

- Befehlskompatibel zu STEP® 5 von Siemens
- Standard AS 511 - Schnittstelle für PGs und OPs
- RUN/STOP-Schalter NR/RE/OR-Schalter
- Status-LEDs für Betriebszustand und Diagnose
- Speichermodul extern (MMC)
- Speicher "on board"
- Netzwerkfähig über Standard Ethernet TCP/IP

**Bestelldaten CPU 24x**

| Typ | Bestellnummer | Beschreibung |
|---|---|---|
| CPU 241 | VIPA 241-1BA01 | SPS CPU 241 mit<br>8KB RAM, 1024 Merker, 32 Timer |
| CPU 242 | VIPA 242-1BA01 | SPS CPU 242 mit<br>32KB RAM, 2048 Merker, 64 Timer,<br>Zeitbearbeitung |
| CPU 243 | VIPA 243-1BA01 | SPS CPU 243 mit<br>52KB RAM, 2048 Merker, 128 Timer,<br>Zeitbearbeitung, Uhr |
| CPU 244 | VIPA 244-1BA01 | SPS CPU 244 mit<br>104KB RAM, 2048 Merker, 8192 S-Merker,<br>128 Timer, Zeit und Alarmbearbeitung, Uhr |

**CPU 24x NET**

Wie CPU 24x zusätzlich mit:

- direktem Anschluss von Twisted Pair über RJ45,
- Datendurchsatz bis zu 100 Telegramme/sec,
- bis zu 20% weniger Busbelastung durch vereinfachtes Quittungsverfahren,
- Treiber für verschiedene Visualisierungssysteme wie zenOn, InTouch usw..

**Bestelldaten CPU 24x NET**

| Typ | Bestellnummer | Beschreibung |
|---|---|---|
| CPU 241 NET | VIPA 241-2BT01 | SPS CPU 241 mit CP243 für H1 / TCP/IP 8KB RAM, 1024 Merker, 32 Timer |
| CPU 242 NET | VIPA 242-2BT01 | SPS CPU 242 mit CP243 für H1 / TCP/IP 32KB RAM, 2048 Merker, 64 Timer, Zeitbearbeitung |
| CPU 243 NET | VIPA 243-2BT01 | SPS CPU 243 mit CP243 für H1 / TCP/IP 52KB RAM, 2048 Merker, 128 Timer, Zeitbearbeitung, Uhr |
| CPU 244 NET | VIPA 244-2BT01 | SPS CPU 244 mit CP243 für H1 / TCP/IP 104KB RAM, 2048 Merker, 8192 S-Merker, 128 Timer, Zeit und Alarmbearbeitung, Uhr |
| CPU 241 NET | VIPA 241-2BT10 | SPS CPU 241 mit CP243 für TCP/IP 8KB RAM, 1024 Merker, 32 Timer |
| CPU 242 NET | VIPA 242-2BT10 | SPS CPU 242 mit CP243 für TCP/IP 32KB RAM, 2048 Merker, 64 Timer, Zeitbearbeitung |
| CPU 243 NET | VIPA 243-2BT10 | SPS CPU 243 mit CP243 für TCP/IP 52KB RAM, 2048 Merker, 128 Timer, Zeitbearbeitung, Uhr |
| CPU 244 NET | VIPA 244-2BT10 | SPS CPU 244 mit CP243 für TCP/IP 104KB RAM, 2048 Merker, 8192 S-Merker, 128 Timer, Zeit und Alarmbearbeitung, Uhr |

**CPU 24x DP**

Wie CPU 24x zusätzlich mit

- integriertem Profibus-Slave
- Status-LEDs für Profibus-Status und -Diagnose
- Lichtwellenleiter-Anschluss (optional)

**Bestelldaten CPU 24x DP**

| Typ | Bestellnummer | Beschreibung |
|---|---|---|
| CPU 241 DP | VIPA 241-2BP01 | SPS CPU 241 mit Profibus-Slave, 8KB RAM, 1024 Merker, 32 Timer |
| CPU 242 DP | VIPA 242-2BP01 | SPS CPU 242 mit Profibus-Slave, 32KB RAM, 2048 Merker, 64 Timer, Zeitbearbeitung |
| CPU 243 DP | VIPA 243-2BP01 | SPS CPU 243 mit Profibus-Slave, 52KB RAM, 2048 Merker, 128 Timer, Zeitbearbeitung, Uhr |
| CPU 244 DP | VIPA 244-2BP01 | SPS CPU 244 mit Profibus-Slave, 104KB RAM, 2048 Merker, 8192 S-Merker, 128 Timer, Zeit und Alarmbearbeitung, Uhr |

**Allgemeines**

Eine CPU ist ein intelligentes Modul. Hier werden Ihre Steuerungsprogramme ausgeführt. Je nachdem wie leistungsfähig Ihr System sein soll, können Sie zwischen vier CPUs wählen. Je leistungsfähiger die gewählte CPU, desto größer ist der Anwenderspeicher.

Diese CPUs 24x sind für kleine und mittlere Anwendungen mit integriertem 24V-Netzteil. Die CPUs enthalten einen Standardprozessor mit internem Programmspeicher sowie ein Flash-ROM für die Speicherung des Anwenderprogramms. Weiter besitzt jede CPU 24x an der Front einen Steckplatz für ein Speicher-Modul.

Jede CPU hat einen PG/OP-Anschluss und ist befehlskompatibel zu STEP® 5 von Siemens. Sie besitzen den Leistungsumfang der 90U bis 115U/944.

Mit dieser CPU-Serie haben Sie Zugriff auf die Peripherie-Module des System 200V. Sie können über standardisierte Befehle und Programme Sensoren abfragen und Aktoren steuern. Sie können maximal 32 Module ansprechen. Über die normierte PG/OP-Schnittstelle haben Sie Zugriff auf andere Geräte.

Im Vergleich zur CPU für die 115U sind die Bedien- und Anzeigeelemente gleich oder ähnlich angeordnet. Das gleiche gilt auch für die Anzeige von Betriebszuständen.

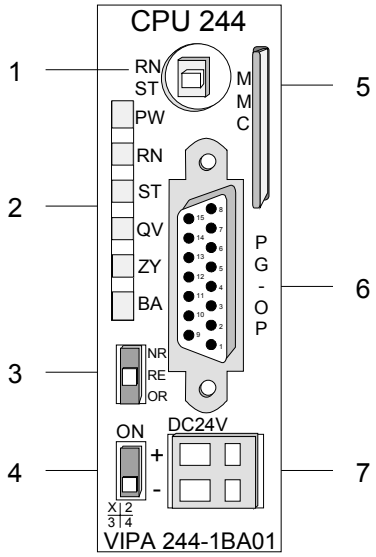
**Die weitere Beschreibung in diesem Kapitel bezieht sich auf die CPU-Familie CPU 24x, da die CPUs 241 bis 244 funktionell gleich sind und sich nur im Speicherausbau unterscheiden.**

**Eigenschaften**

- Befehlskompatibel zu STEP® 5 von Siemens
- Anwenderadressierung über DB1
- Integriertes 24V-Netzteil
- Speicher 8 ... 104 kByte
- Batteriegepufferte Uhr (nur CPU 243 und CPU 244)
- Steckplatz für Memory-Card
- PG/OP-Schnittstelle
- Integrierter VBUS-Kontroller zur Steuerung der Peripherie-Module
- Sicherung des Anwenderprogramms auf internem Flash-ROM
- Integrierte FBs und OBs
- 128 Zeiten
- 128 Zähler
- 2048 Merker
- 8192 S-Merker

# Aufbau

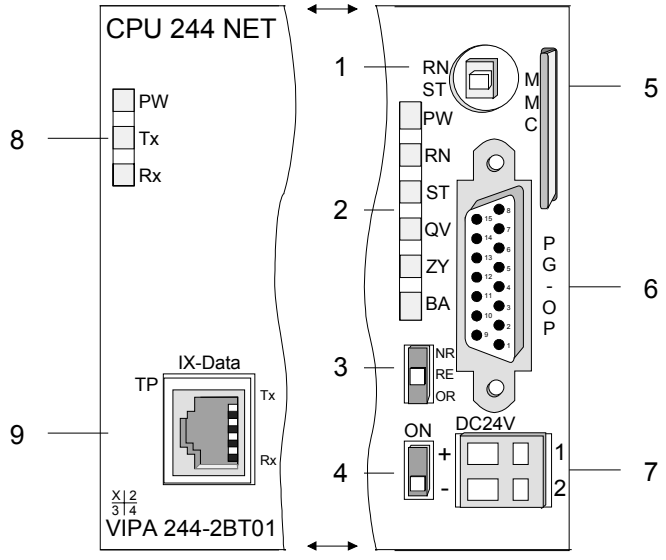
**Frontansicht CPU 24x**




[1] Betriebsarten-Schalter RUN/STOP
[2] LED Statusanzeigen
[3] Voreinstellungsschalter für Verhalten bei Neustart
[4] ON/OFF-Schalter interne Spannungsversorgung
[5] Steckplatz MMC-Speicherkarte
[6] AS 511 - Schnittstelle für PGs und OPs
[7] Anschluss für DC 24V-Spannungsversorgung

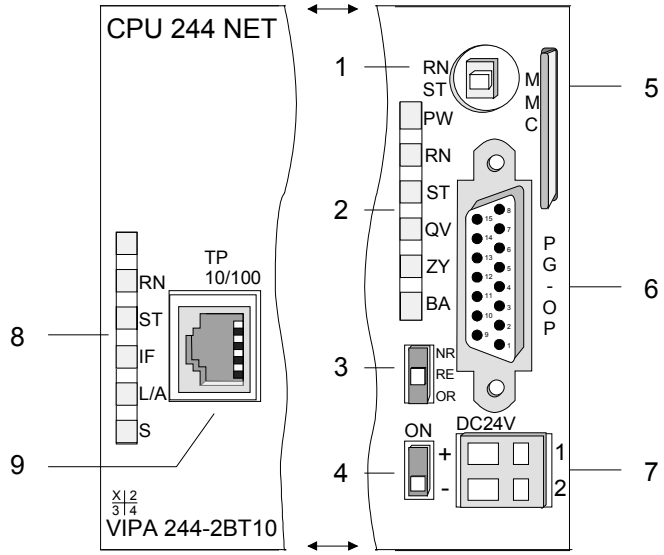
**Frontansicht CPU 24x NET**

CPU 24x-2BT01



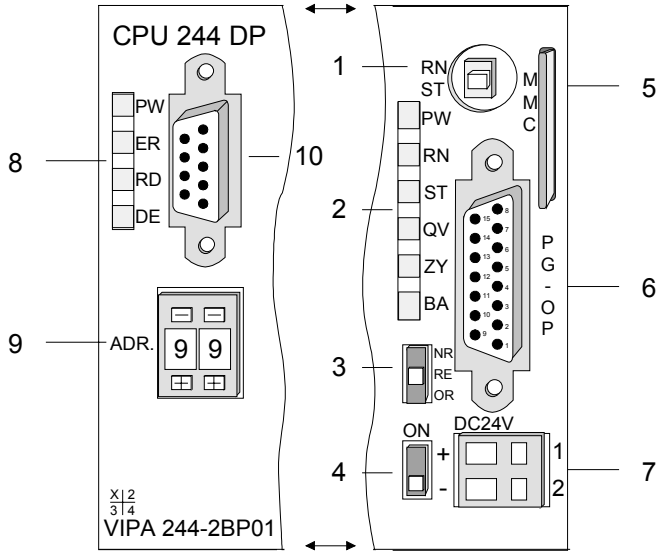

[1] Betriebsarten-Schalter RUN/STOP
[2] LEDs Statusanzeigen
[3] Voreinstellungsschalter für Verhalten bei Neustart
[4] ON/OFF-Schalter interne Spannungsversorgung
[5] Steckplatz MMC-Speicherkarte
[6] AS 511 - Schnittstelle für PGs und OPs
[7] Anschluss für DC 24V-Spannungsversorgung
[8] LEDs Statusanzeige Ethernet
[9] Twisted Pair Schnittstelle für Ethernet

CPU 24x-2BT10




[1] Betriebsarten-Schalter RUN/STOP
[2] LEDs Statusanzeigen
[3] Voreinstellungsschalter für Verhalten bei Neustart
[4] ON/OFF-Schalter interne Spannungsversorgung
[5] Steckplatz MMC-Speicherkarte
[6] AS 511 - Schnittstelle für PGs und OPs
[7] Anschluss für DC 24V-Spannungsversorgung
[8] LEDs Statusanzeige Ethernet
[9] Twisted Pair Schnittstelle für Ethernet

**Frontansicht**
**CPU 24x DP**




[1] Betriebsarten-Schalter RUN/STOP
[2] LEDs Statusanzeigen
[3] Voreinstellungsschalter für Verhalten bei Neustart
[4] ON/OFF-Schalter interne Spannungsversorgung
[5] Steckplatz MMC-Speicherkarte
[6] AS 511 - Schnittstelle für PGs und OPs
[7] Anschluss für DC 24V-Spannungsversorgung
[8] LEDs Statusanzeige Profibus
[9] Einsteller für Profibusadresse
[10] Profibus-Schnittstelle

# Komponenten

**CPU 24x**

Die hier für die CPU 24x beschriebenen Komponenten sind auch Bestandteil aller in diesem Handbuch vorgestellten CPUs.

**LEDs**

Die CPUs 24x besitzen verschiedene LEDs, die der Busdiagnose und der Programm-Statusanzeige dienen. Die Verwendung und die jeweiligen Farben dieser Diagnose-LEDs finden Sie in der nachfolgenden Tabelle.

Diese LEDs sind Bestandteil jeder in diesem Handbuch vorgestellten CPU.

| Bez. | Farbe | Bedeutung |
|---|---|---|
| PW | Grün | Signalisiert die eingeschaltete CPU. |
| RN | Grün | CPU befindet sich in RUN-Betriebszustand. |
| ST | Rot | CPU befindet sich in STOP-Betriebszustand.<br>Wenn beim Einsatz eines Speicher-Moduls eine Ungleichheit zwischen dem Anwenderprogramm auf dem Speicher-Modul und dem im Flash-ROM festgestellt wird, blinkt die rote LED dreimal.<br>Nach dem erstmaligen Stecken kann die rote LED blinken - Abhilfe: Führen Sie ein URLÖSCHEN durch. |
| QV | Rot | Quittungsverzugszeit wurde überschritten. |
| ZY | Rot | Zyklusverzugszeit wurde überschritten. |
| BA | Rot | Befehlsausgabesperre (BASP) ist aktiviert, d.h. die Ausgänge der Ausgabebaugruppen werden nicht freigegeben. |

**Schalter**

**Betriebsarten-Schalter**

Mit dem Betriebsartenschalter können Sie zwischen den Betriebsarten STOP (ST) und RUN (RN) wählen. Die Betriebsart ANLAUF wird von der CPU automatisch zwischen STOP und RUN ausgeführt.

**Voreinstellungs-Schalter**

Mit dem Schalter für Voreinstellungen bestimmen Sie das Verhalten der Zeiten, Zähler und Merker beim Neustart.

Zeiten, Zähler und Merker heißen *remanent* (RE), wenn ihr Inhalt beim Neustart nicht verloren geht.

*Nichtremanent* (NR) sind Zeiten, Zähler und Merker, die beim Neustart zurückgesetzt werden.

Folgendes Remanenzverhalten ist nach dem URLÖSCHEN voreingestellt:

In Schalterstellung NR sind alle Zeiten, Zähler und Merker nichtremanent, in Schalterstellung RE ist jeweils die Hälfte der Zeiten, Zähler und Merker remanent.

**... Fortsetzung Schalter**

**URLÖSCHEN (Overall-RESET)**

Mit der Schalterstellung OR wird **URLÖSCHEN** durchgeführt. Näheres siehe "URLÖSCHEN und Neustart" in diesem Kapitel.

**ON/OFF-Schalter**

Mit dem ON/OFF-Schalter schalten Sie die interne Spannungsversorgung der CPU ein bzw. aus. Bei eingeschalteter Spannungsversorgung werden CPU-Elektronik und Rückwandbus mit den entsprechenden Spannungen versorgt.

**Steckplatz MMC-Speicherkarte**

Als externes Speichermedium können Sie hier ein MMC-Speicher-Modul von VIPA einsetzen (Best.-Nr.: VIPA 241-1XY10).

**Spannungs-versorgung**

Die CPU besitzt ein eingebautes Netzteil. Der Anschluss erfolgt über zwei Anschlussklemmen an der Frontseite. Mit dem ON/OFF-Schalter schalten Sie das Netzteil. In Stellung OFF ist die Versorgung des Rückwandbuses und der CPU-Elektronik unterbrochen.

Das Netzteil ist mit DC 24V (20 ... 30V) zu versorgen. Über die Versorgungsspannung werden neben der CPU-Elektronik auch die angeschlossenen Module über den Rückwandbus versorgt.

Die CPU-Elektronik ist nicht galvanisch von der Versorgungsspannung getrennt. Das Netzteil ist gegen Verpolung und Überstrom geschützt.

**Hinweis!**

Bitte achten Sie auf richtige Polarität bei der Spannungsversorgung.

**Batteriepufferung für Uhr und RAM**

Jede CPU 24x besitzt einen internen Akku, der zur Sicherung des RAMs bei Stromausfall dient.

Zusätzlich wird bei den CPUs 243 und 244 die interne Uhr über den Akku gepuffert.

Der Akku wird direkt über die eingebaute Spannungsversorgung über eine Ladeelektronik geladen und gewährleistet eine Pufferung für max. 30 Tage.

**Achtung!**

Die Pufferbatterie muss in Ordnung sein, damit die CPU in Betrieb gehen kann.

Bei einem Fehler der eingebauten Batterie wird OB 34 aufgerufen.

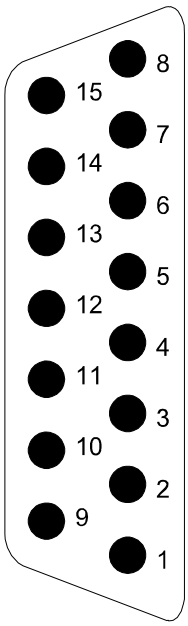
In diesem Fall sollte die CPU überprüft werden. Setzen Sie sich hierzu mit der VIPA in Verbindung!

**PG/OP-Schnittstelle**

An der Frontseite befindet sich eine PG/OP-Schnittstelle. An der PG/OP-Schnittstelle können Sie Programmier- und Bediengeräte anschließen.

Die PG/OP-Schnittstelle ist mit einem Sender und einem Empfänger für 20-mA-Linienstromsignale ausgestattet. Die Datenübertragung der PG/OP-Schnittstelle erfolgt immer mit 9600 Baud.

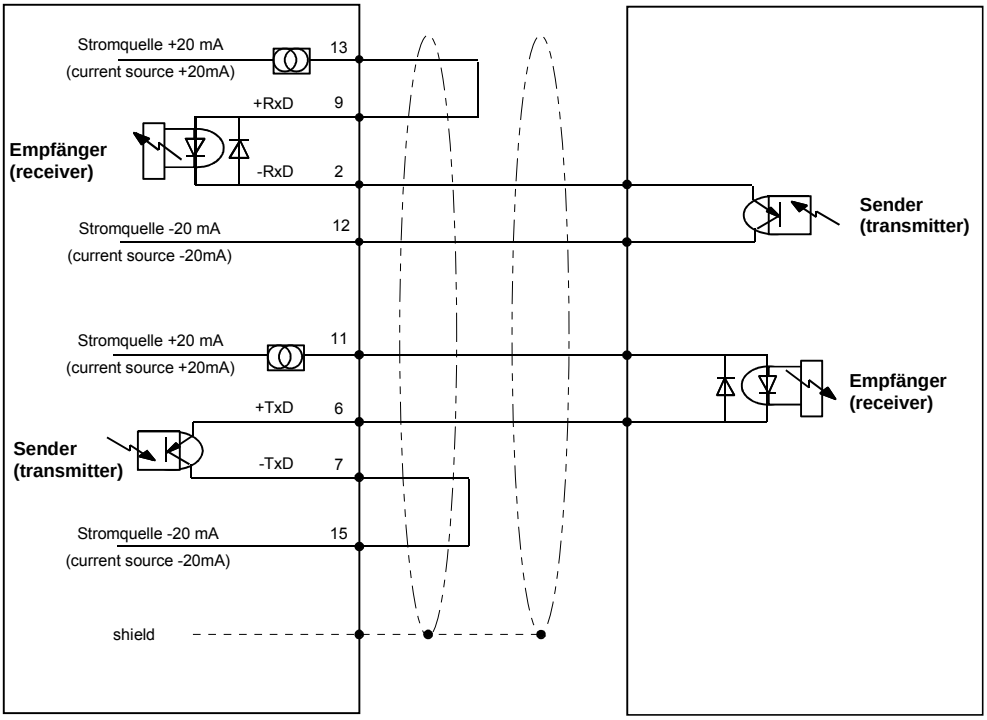
*15pol PG/OP-SubD-Buchse:*



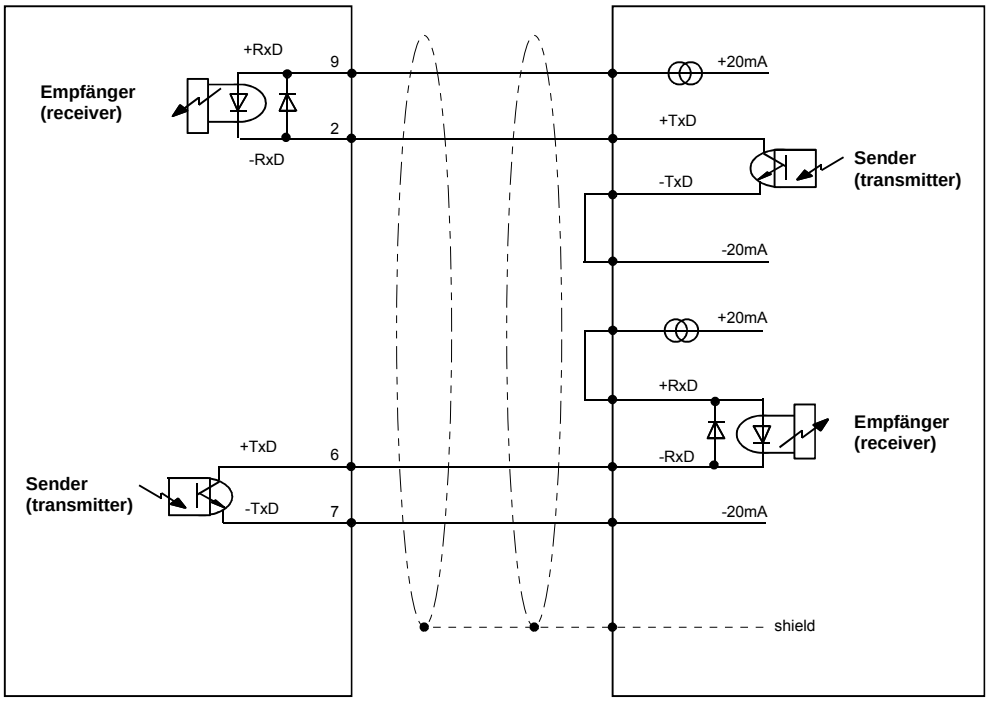

| Pin | Signal | Signalrichtung | Funktion |
|---|---|---|---|
| 1 | Schirm | | Verbunden mit Modulschirm |
| 2 | -RxD | Ausgang | Datenleitung Receive |
| 3 | 5V | Ausgang | Betriebsspannung max. 100mA |
| 4 | 24V | Ausgang | Betriebsspannung max. 100mA |
| 5 | GND | | Modulmasse |
| 6 | +TxD | Eingang | Datenleitung Transmit |
| 7 | -TxD | Ausgang | Datenleitung Transmit |
| 8 | Schirm | | Verbunden mit Modulschirm |
| 9 | +RxD | Eingang | Datenleitung Receive |
| 10 | GND | | Modulmasse |
| 11 | +20mA | Ausgang | Stromquelle 1 |
| 12 | GND | | Modulmasse |
| 13 | +20mA | Ausgang | Stromquelle 2 |
| 14 | 5V | Ausgang | Betriebsspannung |
| 15 | GND | | Modulmasse |

Die 20-mA-Schnittstelle kann aktiv oder passiv betrieben werden. Den für den aktiven Betrieb erforderlichen Strom stellt die Schnittstelle zur Verfügung.

*Aktive 20mA-Schnittstelle*




*Passive 20mA-Schnittstelle*



## CPU 24x NET

Zusätzlich zu den unter CPU 24x erklärten Komponenten besitzt die CPU 24x NET auf der linken Seite 3 weitere LEDs und einen Ethernet-Anschluss, die hier näher beschrieben werden.

### LEDs
CPU 24x-2BT01

Die LEDs befinden sich auf der Front der linken Gehäusehälfte und zeigen anliegende Spannung und Kommunikation an.

Die Verwendung und die jeweiligen Farben dieser LEDs finden Sie in der nachfolgenden Tabelle.

| Bez. | Farbe | Bedeutung |
|---|---|---|
| PW | Grün | Signalisiert eine anliegende Betriebsspannung |
| TxD | Grün | Daten senden (transmit data) |
| RxD | Grün | Daten empfangen (receiv data) |

### LEDs
CPU 24x-2BT10

Der CP besitzt verschiedene LEDs, die der Busdiagnose dienen und den eigenen Betriebszustand anzeigen.

Diese geben nach folgendem Schema Auskunft über den Betriebszustand des CP:

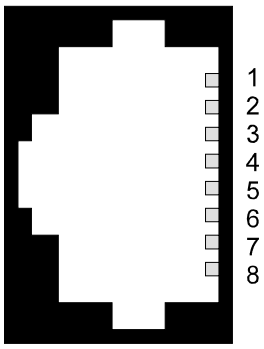
| RUN grün | STOP gelb | SF rot | L/A grün | S grün | Bedeutung |
|---|---|---|---|---|---|
| ○ | ○ | ○ | ○ | ○ | CP wird nicht mit Spannung versorgt, ggf. liegt ein Defekt vor. |
| ● | ○ | ● | X | X | Anlaufphase |
| ● | ○ | ○ | X | X | CP befindet sich mit einem Projekt im RUN befindet und die Kommunikation über projektierbare Verbindungen ist freigegeben. |
| ○ | ● | ○ | X | X | CP befindet sich in STOP, alle projektierbaren Verbindungen sind gesperrt oder CP besitzt kein Projekt, er ist ausschließlich über die MAC-Adresse erreichbar. |
| X | X | X | ● | X | CP ist physikalisch mit Ethernet verbunden |
| X | X | X | ☼ | X | Zeigt Kommunikation über Ethernet (Activity) |
| X | X | X | ○ | ○ | Es besteht keine physikalische Verbindung zum Ethernet. |
| X | X | X | X | ● | Speed: 100MBit |
| X | X | X | X | ○ | Speed: 10MBit |

an: ● aus: ○ blinkend: ☼ irrelevant: X

**Ethernet-Anschluss**

Über die RJ45-Buchse haben Sie einen Twisted-Pair-Anschluss an Ethernet. Die Buchse hat folgende Belegung:

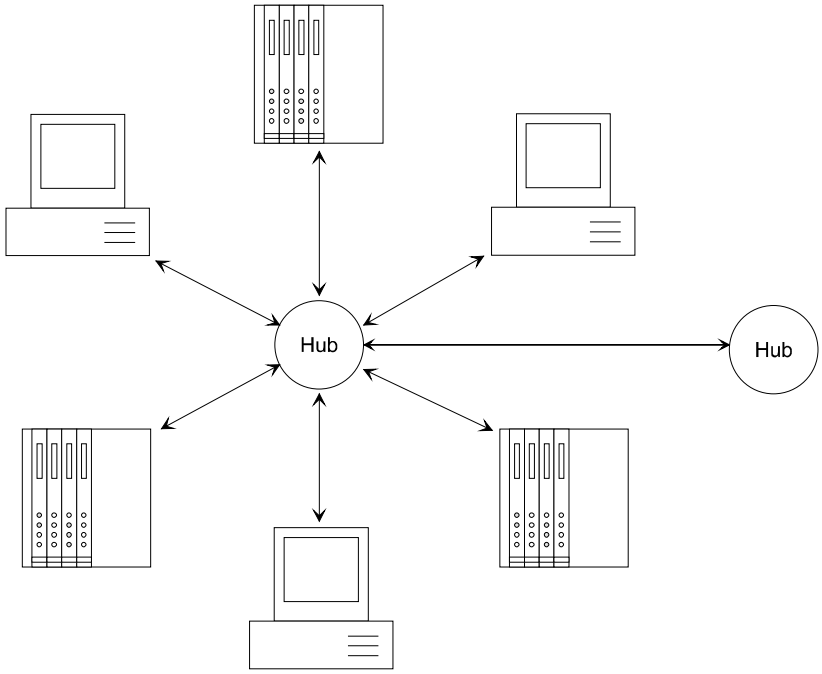
*8-polige RJ45-Buchse:*




| Pin | Signal |
|---|---|
| 1 | Transmit + |
| 2 | Transmit - |
| 3 | Receive + |
| 4 | - |
| 5 | - |
| 6 | Receive - |
| 7 | - |
| 8 | - |

Sterntopologie

Sie können ein Twisted Pair-Netzwerk nur sternförmig aufbauen. Für die Sterntopologie ist ein Sternkoppler erforderlich:



## CPU 24x DP

**LEDs**

Die LEDs befinden sich auf der Front der linken Gehäusehälfte und dienen der Diagnose. Die Verwendung und die jeweiligen Farben dieser LEDs finden Sie in der nachfolgenden Tabelle.

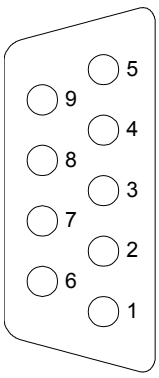
| Bezeichnung | Farbe | Bedeutung |
|---|---|---|
| PW | Grün | Signalisiert eine anliegende Betriebsspannung (Power). |
| ER | Rot | Leuchtet bei Fehler (Error).<br>Leuchtet bei gestoppter CPU.<br>Blinkt langsam (2Hz) bei Initialisierungsfehler.<br>Blinkt schnell (10Hz) wenn Versorgungsspannung unter 18V sinkt.<br>Blinkt abwechselnd mit RD bei fehlerhafter Konfiguration vom Master (Projektierungsfehler).<br>Blinkt gleichzeitig mit RD bei fehlerhafter Parametrierung |
| RD | Grün | Leuchtet bei Datenübertragung über den V-Bus.<br>Blinkt bei positivem Selbsttest (READY) und erfolgreicher Initialisierung. |
| DE | Gelb | DE (Data exchange) zeigt an, dass eine Kommunikation mit dem Profibus stattfindet. |

**Profibus-Anschluss**

Die CPU 24x DP wird über eine 9-polige Buchse in das Profibus-System eingebunden.

Die Anschlussbelegung dieser Schnittstelle zeigt folgende Abbildung:

*9-polige Profibus-SubD-Buchse:*

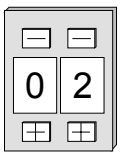



| Pin | Belegung |
|---|---|
| 1 | Schirm |
| 2 | n.c. |
| 3 | RxD/TxD-P |
| 4 | CNTR-P |
| 5 | GND |
| 6 | 5V (max. 70mA) |
| 7 | n.c. |
| 8 | RxD/TxD-N |
| 9 | n.c. |

**Adresseinsteller**

Über den Adresseinsteller stellen Sie die Adresse ein, die zur Projektierung für den Bus-Koppler verwendet werden soll.

Erlaubte Adressen sind 1 bis 99. Jede Adresse darf nur einmal am Bus vergeben sein. Die Slave-Adresse muss vor dem Einschalten des Buskopplers eingestellt werden.




**Achtung!**

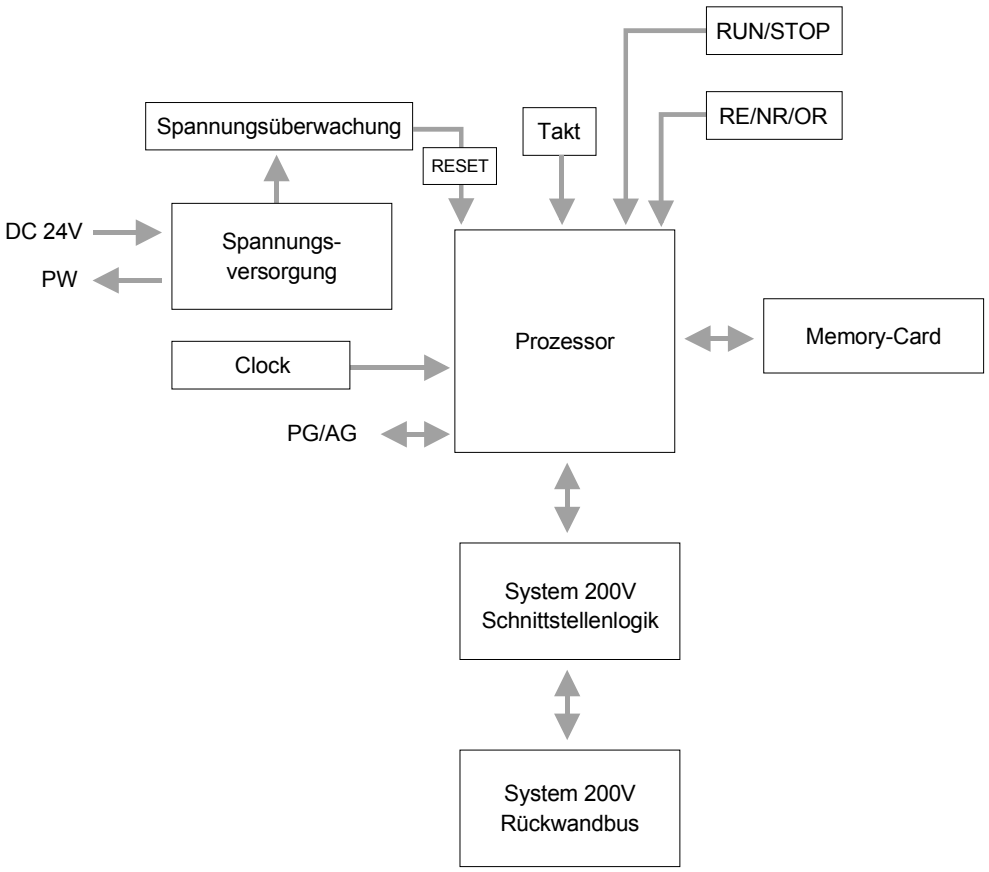
Im laufenden Betrieb darf die Adresse nicht geändert werden!

**Hinweis!**

Näheres zum Thema Profibus finden Sie im Teil "Einsatz CPU 24x DP".

# Blockschaltbild

Das nachfolgende Blockschaltbild zeigt den prinzipiellen Hardwareaufbau der CPU 24x-Module:

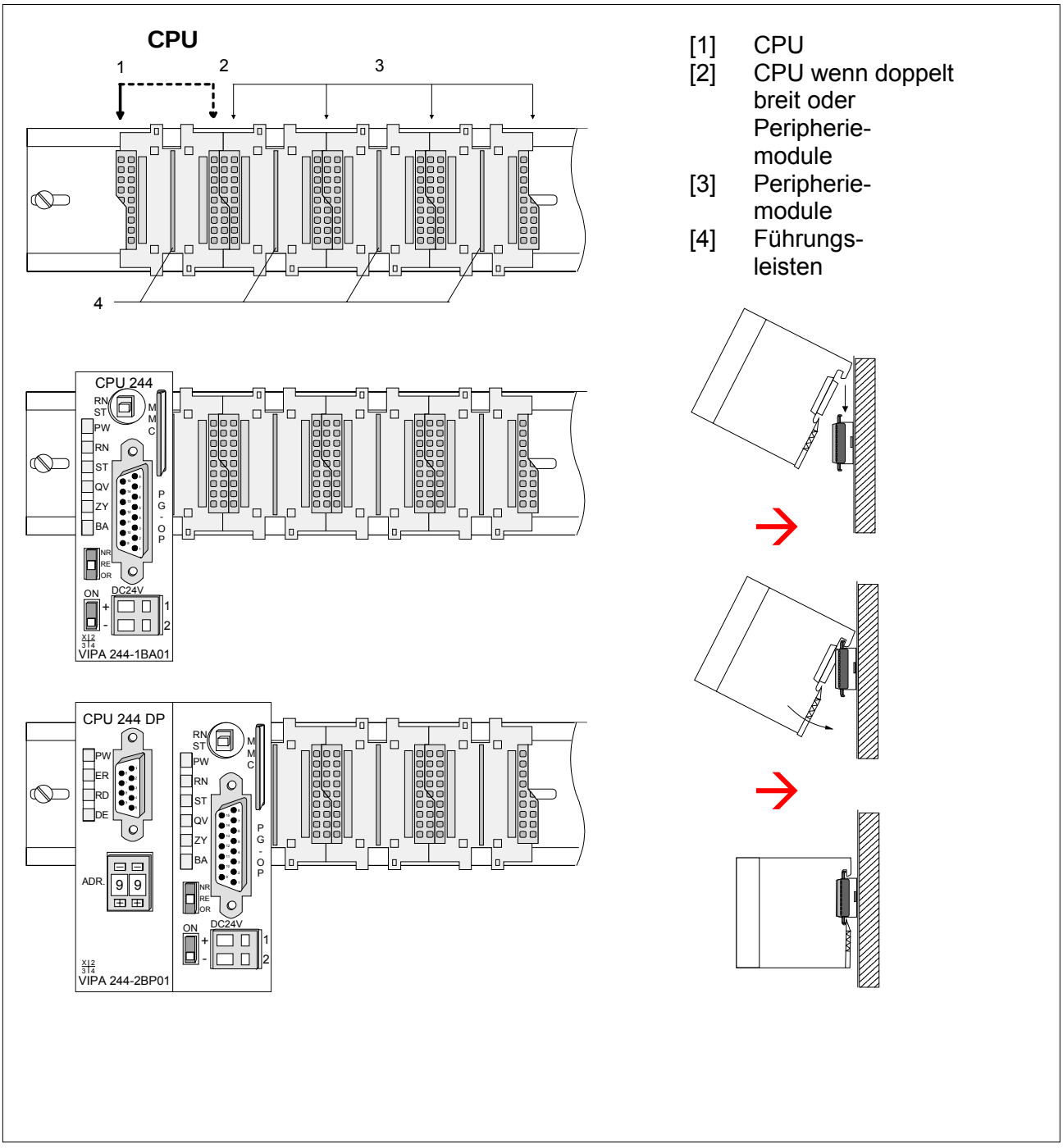


# Inbetriebnahme

**Achtung!**
Schalten Sie die Stromversorgung aus, bevor Sie Module stecken bzw. abziehen!

**Montage**

Bitte beachten Sie, dass Sie die CPU nur auf Steckplatz 1 bzw. 2 stecken dürfen (siehe nachfolgende Abbildung).



# Anlaufverhalten

**Anlauf**

Die Vorgänge zwischen STOP-RUN-Übergang (manueller Neustart) und NETZ-EIN-RUN-Übergang nennt man Anlaufverhalten.

Nach dem Einschalten ermittelt der Prozessor die am Rückwand-Bus befindlichen Module und speichert die entsprechenden Werte ab. Es leuchten die LEDs BA, RN und ST.

Während die CPU sich im Anlauf befindet wird der entsprechende Anlauf-OB bearbeitet:

- bei manuellem Neustart: OB 21,
- bei NETZ EIN und Schalterstellung RN: OB 22.

**Achtung!**

Die Baugruppen müssen auf den richtigen Steckplätzen im System 200V stecken.

Die Pufferbatterie muss in Ordnung sein, damit die CPU in Betrieb gehen kann.

**Anwender-programm automatisch nachladen**

Nach einem NETZ EIN wird automatisch das Anwenderprogramm aus dem Flash-ROM oder bei gestecktem Speicher-Modul aus dem Speicher-Modul in das RAM übertragen und in das Flash-ROM geschrieben.

Voraussetzung hierfür:

- Das Anwenderprogramm im Flash-ROM bzw. auf dem Speicher-Modul ist gültig.
- Das Nachladen nach NETZ EIN ist aktiviert.

Näheres hierzu siehe weiter unten unter "Anwenderprogramm laden und sichern".

# URLÖSCHEN und Neustart

**URLÖSCHEN**

Beim URLÖSCHEN wird der komplette Anwenderspeicher (RAM) gelöscht. Die Daten im Flash-ROM bleiben erhalten.

Zum URLÖSCHEN gehen Sie folgendermaßen vor:

- Stellen Sie den Betriebsartenschalter auf "STOP".
- Schalten Sie die Netzspannung ein.

  → Folgende LEDs müssen auf der CPU leuchten:
- Rote LED "STOP"
- Rote LED "BASP".
- Halten Sie den Betriebsartentaster in Stellung "OR" fest, und stellen Sie gleichzeitig den Betriebsartenschalter von "ST" nach "RN".
- Diesen Vorgang mehrmals wiederholen. Dabei erlischt die rote LED "ST" kurz.

  → Die CPU ist urgelöscht und die rote LED "ST" zeigt Dauerlicht.

**Neustart**

- Lassen Sie den Betriebsartentaster los.
- Stellen Sie den Betriebsartenschalter auf "ST" und wieder auf "RN".

  → Rote LED "STOP" erlischt,
  Grüne LED "RUN" leuchtet,
  Rote LED "BASP" erlischt.

Solange die CPU die Neustart-Routine bearbeitet:

- leuchtet die LED "BASP"; und "RUN" "STOP" gleichzeitig,
- sind alle Ausgabebaugruppen gesperrt, Ausgänge führen Signal "0",
- führen alle Ein- und Ausgänge im Prozessabbild Signal "0",
- ist die Zyklusüberwachungszeit inaktiv.

Sobald die BASP-LED erlischt, befindet sich die CPU im RUN.

**Programm nach URLÖSCHEN nachladen**

Nach URLÖSCHEN wird automatisch das Anwenderprogramm aus dem Flash-ROM in das RAM übertragen wenn:

- das Anwenderprogramm im Flash-ROM gültig ist,
- das Nachladen nach URLÖSCHEN aktiviert ist.

Näheres hierzu siehe auf der Folgeseite unter "Anwenderprogramm laden und sichern".

# Anwenderprogramm laden und sichern

**Flash-ROM und MMC**

Die CPU bietet die Möglichkeit Ihr Anwenderprogramm zu laden und zu sichern:

- im internen Flash-ROM,
- extern auf Speicher-Modul (MMC).

Die Sicherung wird über die Systemdatenzelle BS 38 koordiniert. Es sind folgende Einstellungen möglich:

| Einstellungen | **BS 38 Datum** | Erläuterung |
|---|---|---|
| Sicherung im Flash-ROM und auf der MMC. | **0001h** | Das Anwenderprogramm wird im Flash-ROM und wenn gesteckt auf der MMC gesichert. |
| Sicherung im Flash-ROM und auf der MMC.<br>Nachladen nach NETZ EIN von MMC bzw. Flash-ROM. | **0003h** | Das Anwenderprogramm wird im Flash-ROM und wenn gesteckt auf der MMC gesichert .<br>Das Programm wird nach NETZ EIN bei gesteckter MMC von der MMC geladen oder bei nicht gesteckter MMC vom Flash-ROM nachgeladen. |
| Sicherung im Flash-ROM und auf der MMC.<br>Nachladen nach URLÖSCHEN nur von Flash-ROM. | **0005h** | Das Anwenderprogramm wird im Flash-ROM gesichert. Das Programm wird beim URLÖSCHEN vom Flash-ROM nachgeladen. |
| Sicherung im Flash-ROM und auf der MMC.<br>Nachladen nach NETZ EIN von MMC bzw. Flash-ROM und nachladen nach URLÖSCHEN nur von Flash-ROM. | **0007h** | Das Anwenderprogramm wird im Flash-ROM gesichert. Das Programm wird nach NETZ EIN und beim URLÖSCHEN nachgeladen. |
| Anwenderprogramm im Flash-ROM und auf der MMC für ungültig erklären. | **FFFFh** | Ein in Flash-ROM bzw. MMC vorhandenes Programm wird gelöscht.<br>Dieser Vorgang kann nicht mehr rückgängig gemacht werden |

Die Sicherung erfolgt beim STOP/RUN Übergang. Der Inhalt der BS Zelle 38 wird nach der Sicherung auf Null gesetzt.

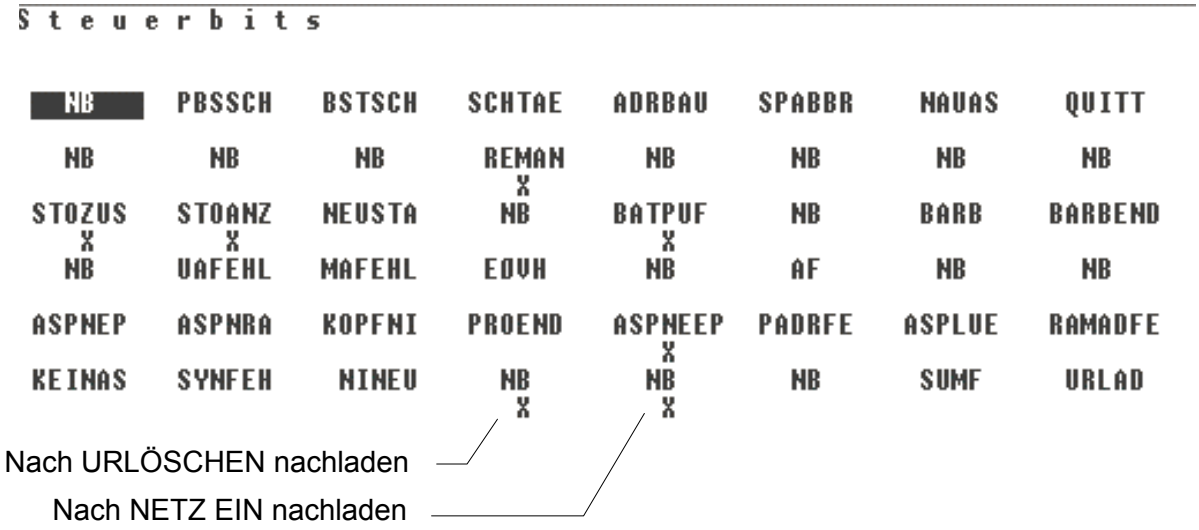
**Einstellungen anzeigen**

Nach der Sicherung des Programms sind die Einstellungen im BS 7 bzw. im U-Stack sichtbar.

*Kontrolle in BS 7*

| BS 7 Bit Nr. | Bezeichnung im U-Stack | Bedeutung |
|---|---|---|
| 3 | NB | Programm bei jedem NETZ EIN von MMC bzw. Flash-ROM laden. |
| 4 | NB | Programm nach URLÖSCHEN vom Flash-ROM laden. |

*Kontrolle in U-Stack*


Steuerbits
NB PBSSCH BSTSCH SCHTAE ADRBAU SPABBR NAUAS QUITT
NB NB NB REMAN NB NB NB NB
X
STOZUS STOANZ NEUSTA NB BATPUF NB BARB BARBEND
X X X
NB UAFEHL MAFEHL EOVH NB AF NB NB
ASPNEP ASPNRA KOPFNI PROEND ASPNEEP PADRFE ASPLUE RAMADFE
X
KEINAS SYNFEH NINEU NB NB NB SUMF URLAD
X X
Nach URLÖSCHEN nachladen
Nach NETZ EIN nachladen

# Testfunktion STATUS/STATUS VAR und STEUERN

Mit der Testfunktion STATUS und STATUS VAR können die Signalzustände von Operanden und das VKE angezeigt werden.

Bei dem programmabhängigen STATUS werden die Signalzustände während des Zyklus angezeigt. Mit STATUS VAR erfolgt die direkte Signalzustandsanzeige am Ende des Zyklus.

**STATUS**

Diese Testfunktion zeigt die aktuellen Signalzustände und das VKE der einzelnen Operanden während der Programmbearbeitung an.

Es können außerdem Korrekturen am Programm durchgeführt werden.

**Hinweis!**

Das AG muss bei der Testfunktion "STATUS" in der Betriebsart RUN sein.

Die Statusbearbeitung kann durch Sprungbefehle oder Zeit- und Prozessalarme unterbrochen werden. Die CPU hört an der Unterbrechungsstelle auf, Daten für die Statusanzeige zu sammeln und übergibt dem PG anstelle der noch benötigten Daten nur Daten mit dem Wert 0.

Deshalb kann es bei Verwendung von Sprungbefehlen oder von Zeit- und Prozessalarmen vorkommen, dass in der Statusanzeige eines Bausteins während dieser Programmbearbeitung nur der Wert 0 angezeigt wird für:

- das Verknüpfungsergebnis VKE
- Status / AKKU 1
- AKKU 2
- Zustandsbyte
- absolute Speicheradresse SAZ. Hinter SAZ erscheint dann ein "?".

Die Unterbrechung der Statusbearbeitung hat keinen Einfluss auf die Programmbearbeitung, sondern macht nur deutlich, dass die angezeigten Daten ab der Unterbrechungsstelle nicht mehr gültig sind.

**STATUS VAR**

Diese Testfunktion gibt den Zustand eines beliebigen Operanden (Eingänge, Ausgänge, Merker, Datenwort, Zähler oder Zeiten) am Ende einer Programmbearbeitung an.

Diese Informationen werden aus dem Prozessabbild der ausgesuchten Operanden entnommen. Während der "Bearbeitungskontrolle" oder in der Betriebsart STOP wird bei den Eingängen direkt die Peripherie eingelesen. Andernfalls wird nur das Prozessabbild der aufgerufenen Operanden angezeigt.

**STEUERN**

**STEUERN von Ausgängen**

Dadurch kann die Verdrahtung und die Funktionstüchtigkeit von Ausgabebaugruppen kontrolliert werden.

Auch ohne Steuerungsprogramm können Ausgänge auf den gewünschten Signalzustand eingestellt werden. Das Prozessabbild wird dabei nicht verändert, die Sperre der Ausgänge jedoch aufgehoben.

**Hinweis!**

Das AG muss bei dieser Testfunktion in der Betriebsart STOP sein.

Die Funktion sollte nur ohne Lastspannung der Verbraucher durchgeführt werden.

**STEUERN VAR**

**STEUERN von Variablen**

Folgende Variablen können geändert werden:

E, A, M, T, Z, und D.

Unabhängig von der Betriebsart des AGs wird das Prozessabbild binärer und digitaler Operanden verändert.

In der Betriebsart RUN wird die Programmbearbeitung mit den geänderten Prozessvariablen ausgeführt. Im weiteren Programmablauf können sie jedoch, ohne Rückmeldung wieder verändert werden.

Die Prozessvariablen werden asynchron zum Programmablauf gesteuert.

Besonderheiten:

- Die Variablen E, A und M nur byte- oder wortweise im Prozessabbild verändern.
- Bei den Variablen T und Z im Format KM und KH
  zusätzlich in der Maske VOREINSTELLUNGEN im Eingabefeld SYSTEMBEFEHLE ein "JA" eingeben,
  die Steuerung der Flankenmerker beachten.
- Die Signalzustandsanzeige wird abgebrochen, falls eine fehlerhafte Format- oder Operandeneingabe vorliegt. Das PG gibt dann die Meldung "KEIN STEUERN MÖGLICH" aus.

# MMC-Speicher-Modul

Als externes Speicher-Medium kommt die Multi Media Card (MMC) von VIPA zum Einsatz. Das Speicher-Modul kann bei VIPA bezogen werden und hat die Best.-Nr.: VIPA 241-1XY10.

Im Einsatz mit der CPU 24x wird das Speicher-Modul wie das interne Flash-ROM der CPU 24x behandelt.

**Transfer CPU → MMC**

Bei gesteckter MMC wird bei einem Schreibbefehl das Anwenderprogramm sowohl im Flash-ROM als auch in der MMC gespeichert. Das Schreiben wird über die Systemdatenzelle BS 38 koordiniert (Siehe "Anwenderprogramm laden und sichern").

Bei einem Schreibvorgang erfolgt die Ausgabe eines Lauflichts auf den roten LEDs der CPU.

War die Übertragung erfolgreich, wird im USTACK das Bit ASPNEP (Speichermodul ist EPROM - BS 7 Bit 15) gesetzt.

**Transfer MMC → CPU**

Das Übertragen des Anwenderprogramms von der MMC in die CPU kann nur nach einem NETZ EIN erfolgen.

Die Lauflichtausgabe der roten LEDs der CPU kennzeichnet den Übertragungsvorgang.

Nach der Übertragung in das RAM wird das Anwenderprogramm in das Flash-ROM übertragen.

Hierbei blinkt die rote STOP-LED dreimal. Nach einer erfolgreichen Übertragung von der MMC in die CPU blinkt die grüne RUN-LED dreimal.

Ist kein gültiges Anwenderprogramm auf der gesteckten MMC, oder scheitert die Übertragung, so erfolgt das URLÖSCHEN der CPU. In diesen Fällen wird das Anwenderprogramm im Flash-ROM nicht verändert.

**Hinweis!**

Es ist zu beachten, dass die Speichergröße auf die CPU abgestimmt ist!

Eine mit einer CPU 244 beschriebene MMC kann auch nur von einer CPU 244 ausgelesen werden.

Anwenderprogramme, die von einer CPU 241, 242 oder 243 erstellt und auf die MMC geschrieben wurden, können untereinander ausgetauscht werden. Hierbei wird die Programmgröße beim Einspielen in die CPU überprüft.

Ist das Anwenderprogramm größer als der Anwenderspeicher in der CPU, wird der Inhalt der MMC nicht in die CPU übertragen.

Vor dem Übertragen des Anwenderprogrammes in das interne Flash bzw. die MMC ist es ratsam eine Komprimierung durchzuführen.

# Speicherbereiche

Nachfolgende Tabelle zeigt die Adressraumbelegung der CPU 24x. In den hier angegebenen Adresszellen ist die Anfangsadresse der Bereiche abgelegt. Für die CPUs 241 ... 243 können die Werte in den Adresszellen direkt als Anfangsadresse verwendet werden.

**Hinweis!**

Systembedingt müssen für die CPU 244 zur Ermittlung der gesuchten Baustein-Adresse die Werte in den Adress-Zellen verdoppelt werden.

Eine Speicherzelle beinhaltet z.B. 0008h. Die richtige Adresse erhalten Sie, indem Sie diesen Wert verdoppeln, d.h. 0008h verdoppelt ergibt 0010h.

| Bereich / CPU | CPU241 | | CPU242 | | CPU243 | | CPU244 | |
|---|---|---|---|---|---|---|---|---|
| | Adresse | Länge | Adresse | Länge | Adresse | Länge | Adresse | Länge |
| Anwenderspeicher | 0000h<br>1FFFh | 2000h<br>(8k) | 0000h<br>1FFFh | 8000h<br>(32k) | 0000h<br>1FFFh | D000h<br>(52k) | 00000h<br>19FFFh | 1A000h<br>(104k) |
| Bausteinadressliste | DC00h | | DC00h | | DC00h | | 1CC00h | |
| OB1 | DC02h | | DC02h | | DC02h | | 1CC02h | |
| FB1 | DE02h | | DE02h | | DE02h | | 1CE02h | |
| PB1 | E002h | | E002h | | E002h | | 1D002h | |
| SB1 | E202h | | E202h | | E202h | | 1D202h | |
| DB1 | E402h | | E402h | | E402h | | 1D402h | |
| FX1 | E602h | | E602h | | E602h | | 1D602h | |
| DX1 | E802h | | E802h | | E802h | | 1D802h | |
| Systemdaten BS | EA00h | | EA00h | | EA00h | | 1DA00h | |
| Zeitfunktionen T | EC00h | | EC00h | | EC00h | | 1DC00h | |
| Zählfunktionen | ED00h | | ED00h | | ED00h | | 1DD00h | |
| Merker | EE00h | | EE00h | | EE00h | | 1DE00h | |
| Prozessabbilder | | | | | | | | |
| PAE | EF00h | | EF00h | | EF00h | | 1DF00h | |
| PAA | EF80h | | EF80h | | EF80h | | 1DF80h | |
| S-Merker | - | | - | | - | | - | |
| | Hinweis! S-Merker sind nur direkt adressierbar! | | | | | | | |

**Adressen im Systemdaten-bereich**

In der nachfolgenden Tabelle finden Sie eine Auswahl der für Sie wichtigen Systemdaten in geordneter Reihenfolge. Die Adressen, die Sie hier sehen sind der Offset zur jeweiligen Systemdatenanfangsadresse der CPUs.

Die Anfangsadressen der Systemdaten BS der CPUs 24x lauten:

| | |
|---|---|
| CPU 241 ... CPU 243: | EA00h |
| CPU 244: | 1DA00h |

| System-datenwort | Offset | Bedeutung |
|---|---|---|
| 8 ... 12 | 10 ... 19 | Uhr: Uhrendatenbereich, Statuswort, Fehlermeldungen, Korrekturwert |
| 32 ... 33 | 40 ... 43 | Anfangsadresse des internen RAMs |
| 34 ... 35 | 44 ... 47 | Endadresse des internen RAMs |
| 36 ... 37 | 48 ... 4B | Füllstandsanzeige des internen RAMs |
| 38 | 4C ... 4D | Koordination der Sicherung im Flash-ROM |
| 40 ... 45 | 50 ... 5B | CPU-Bezeichnung und Firmwarestand in ASCII-Codierung |
| 46 | 5C ... 5D | Treiber-Nr. und Fehlermeldung (z.B. bei ASCII-Treiber) |
| 48 ... 55 | 60 ... 6F | Treiber-Parameterblock (z.B. für ASCII-Treiber) |
| 57 ... 63 | 72 ... 7F | L1-Parameterfeld |
| 64 ... 79 | 80 ... 9F | Ausgangs-Koppelmerker |
| 80 ... 95 | A0 ... BF | Eingangs-Koppelmerker |
| 96 | C0 ... C1 | Zykluszeitüberwachung |
| 97 | C2 ... C3 | Zeitintervall für OB13 (in 10ms 0...65535) |
| 98 | C4 ... C5 | Zeitintervall für OB12 (in 10ms 0...65535) |
| 99 | C6 ... C7 | Zeitintervall für OB11 (in 10ms 0...65535) |
| 100 | C8 ... C9 | Zeitintervall für OB10 (in 10ms 0...65535) |
| 101 | CA ... CB | für OB 6 gestartete Weckzeit (in ms) |
| 102 | CC ... CD | Fehler beim Kopieren des Speichermoduls oder interne Fehlernr. [2]) |
| 103 ... 104 | CE ... D1 | Adresse der fehlerhaften Baugruppe bei QVZ oder Fehleradresse beim Adresslistenaufbau |
| 120 | F0 ... F1 | Systemeigenschaften:<br>Softwareschutz, PAE/PAA lesen/schreiben sperren,<br>Remanenzverhalten von Merkern, Zählern und Zeiten,<br>Prozessabbildtransfer parallel |
| 121 | F2 ... F3 | aktuelle Zykluszeit |
| 122 | F4 ... F5 | maximale Zykluszeit |
| 123 | F6 ... F7 | minimale Zykluszeit |
| 126 | FC ... FD | Anlaufverzögerung (in ms) |
| 128 ... 159 | 100 ... 13F | Liste alle ansprechbaren Peripherieworte |
| 160 ... 175 | 140 ... 15F | Fehlerpuffer für Systemfehlerbehandlung |
| 203 ... 229 | 196 ... 1CB | Unterbrechungsstack |

# USTACK - Ausgabe der Systemdaten mit dem PG

Bits, die eine Störungsursache anzeigen und der Step-Adresszähler sind gekennzeichnet.

| Bit Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | absolute Adr. | System Daten-wort |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | BST SCH | SCH TAE | ADR BAU | SPA BBR | | | EA0A | SD 5 |
| 2 | CA-DA | CE-DA | | REM AN | | | | | EA0B | |
| 3 | STO ZUS | STO ANZ | NEU STA | | BAT PUF | | BARB | BARB END | EA0C | SD 6 |
| 4 | | UA FEHL | | | | AF | | | EAOD | |
| 5 | ASP NEP | ASP NRA | KOPF NI | | ASP NEEP | | | | EA0E | SD 7 |
| 6 | KEIN AS | SYN FEH | NINEU | [2]) BS 7 | [2]) BS 7 | [2]) BS 7 | | UR LAD | EA0F | |
| 7 | irrelevant | | | | | | | | | |
| 8 | irrelevant | | | | | | | | | |
| 9 | STOPS | | SUF | TRAF | NNN | STS | STUEB | FEST | EBAC | SD 214 |
| 10 | NAU | QVZ | KOLIF | ZYK | SYSFE | PEU | BAU | ASPFA | EBAD | (UAW) |
| 11 | | | | | | | | | EBAA | SD 213 |
| 12 | ANZ1 | ANZ0 | OVFL | | OR | STA TUS | VKE | ERAB | EBAB | |
| 13 | 6. Klammerebene | | | | | OR | VKE | FKT | EBA8 | SD 212 |
| 14 | irrelevant | | | | | | | | EBA9 | |
| 15 | 4. Klammerebene | | | | | OR | VKE | FKT | EBA6 | SD 211 |
| 16 | 5. Klammerebene | | | | | OR | VKE | FKT | EBA7 | |
| 17 | 2. Klammerebene | | | | | OR | VKE | FKT | EBA4 | SD 210 |
| 18 | 3. Klammerebene | | | | | OR | VKE | FKT | EBA5 | |
| 19 | Klammertiefe (0...6) | | | | | | | | EBA2 | SD 209 |
| 20 | 1. Klammerebene | | | | | OR | VKE | FKT | EBA3 | |
| 21 | Anfangsadresse des Datenbausteins (high) | | | | | | | | EBA0 | SD 208 |
| 22 | Anfangsadresse des Datenbausteins (low) | | | | | | | | EBA1 | |
| 23 | Bausteinstackpointer (high) | | | | | | | | EB9E | SD 207 |
| 24 | Bausteinstackpointer (low) | | | | | | | | EB9F | |

*Fortsetzung ...*

*... Fortsetzung*

| Bit Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | absolute Adr. | System Daten-wort |
|---|---|---|---|---|---|---|---|---|---|---|
| 25 | Step-Adresszähler (high) | | | | | | | | EB9C | SD 206 |
| 26 | Step-Adresszähler (low) [1] | | | | | | | | EB9D | |
| 27 | Befehlregister (high) | | | | | | | | EB9A | SD 205 |
| 28 | Befehlsregister (low) | | | | | | | | EB9B | |
| 29 | AKKU 2 (high) | | | | | | | | EB98 | SD 204 |
| 30 | AKKU 2 (low) | | | | | | | | EB99 | |
| 31 | AKKU 1 (high) | | | | | | | | EB96 | SD 203 |
| 32 | AKKU 1 (low) | | | | | | | | EB97 | |

[1]) Absolute Speicheradresse der nächsten, noch nicht bearbeiteten Anweisung.

[2]) USTACK BS 7

| BS 7 Bit Nr. | Bezeichnung im U-Stack | Bedeutung |
|---|---|---|
| 2 | NB | Sammelfehler Bit: wird gesetzt bei SYSFE oder PEU oder wenn ein interner Systemfehler aufgetreten ist. |
| 3 | NB | Programm bei jedem NETZ EIN von MMC bzw. Flash-ROM laden. |
| 4 | NB | Programm nach URLÖSCHEN vom Flash-ROM laden. |

**Hinweis!**

Bei internem Systemfehler die BS 102, 103, 110 bis 119 auslesen und der VIPA-Hotline melden.

**Abkürzungen für Steuerbits**

| Abkürzungen für Steuerbits | |
|---|---|
| BSTSCH | Bausteinschieben angefordert |
| SCHTAE | Bausteinschieben aktiv Funktion: KOMP:AG |
| ADRBAU | Adresslistenaufbau |
| SPABBR | Komprimieren abgebrochen |
| CA-DA | Koppelmerkerausgabeadressliste vorhanden |
| CE-DE | Koppelmerkereingabeadressliste vorhanden |
| REMAN | 0: bei Neustart werden alle Zeiten/Zähler und Merker gelöscht<br>1: bei Neustart wird die 2. Hälfte der Zeiten, Zähler und Merker gelöscht |
| STOZUS | STOP-Zustand (externe Anforderung z. B. über PG) |
| STOANZ | STOP-Anzeige |
| NEUSTA | AG im Neustart |
| BATPUF | Batteriepufferung in Ordnung |
| BARB | Bearbeitungskontrolle |
| BARBEND | Bearbeitungskontrolle-Ende-Anforderung |
| UAFEHL | Fehlerhafte Unterbrechungsanzeige |
| AF | Alarmfreigabe |
| ASPNEP | Speichermodul ist EPROM |
| ASPNRA | Speichermodul ist RAM |
| KOPFNI | Bausteinkopf nicht interpretierbar |
| ASPNEEP | Speichermodul ist EEPROM |
| KEINAS | Kein Speichermodul |
| SYNFEH | Synchronisierfehler (Bausteine sind nicht in Ordnung) |
| NINEU | Neustart nicht möglich |
| URLAD | Urladen erforderlich |
| Sonstige Abkürzungen:<br>SD | Systemdaten<br>(ab Adresse EA00h) |

**Abkürzungen für Störungsursachen**

| Abkürzungen für Störungsursachen bzw. Fehlerkennungen | |
|---|---|
| STOPS | Betriebsartenschalter auf STOP |
| SUF | Substitutionsfehler |
| TRAF | Transferfehler bei DB Befehlen: DW-Nr. > Datenbausteinlänge |
| NNN | Befehl in CPU 24x nicht interpretierbar |
| STS | Unterbrechung des Betriebes durch PG-Stop-Anforderung oder programmierter Stop-Anweisungen |
| STUEB | Bausteinstacküberlauf: Die max. Bausteinaufrufverschachtelung von 32 wurde überschritten |
| FEST | Fehler in der Selbsttestroutine der CPU |
| NAU | Netzausfall |
| QVZ | Quittungsverzug von der Peripherie: Es wurde eine nicht vorhandene Baugruppe angesprochen. |
| KOLIF | Koppelmerkertransferliste ist fehlerhaft |
| ZYK | Zykluszeitüberschreitung: Es wurde die eingestellte max. zulässige Programmbearbeitungszeit überschritten. |
| SYSFE | Fehler im DB 1 |
| PEU | Erweiterungsbaugruppe ohne Anschluss<br>Peripheriebus gestört<br>unbekanntes Modul<br>falsch platzierte Baugruppe |
| BAU | Batterieausfall |
| ASPFA | Unzulässiges Speichermodul (wird zur Zeit nicht verwendet) |

**Sonstige Abkürzungen**

| Sonstige Abkürzungen | |
|---|---|
| UAW | Unterbrechungsanzeigewort |
| ANZ1/ANZ0 | 00:AKKU1 = 0 oder 0 geschoben<br>01: AKKU1 > 0 oder 1 geschoben<br>10: AKKU 1 < 0 |
| OVFL | arithmetischer Überlauf (+ oder -) |
| ODER(OR) | ODER-Speicher<br>(gesetzt durch Operation "O") |
| STATUS | STATUS des Operanden der zuletzt ausgeführten Binäroperation. |
| VKE | Verknüpfungsergebnis |
| ERAB | Erstabfrage |
| KE1...KE6 | Klammerstack-Eintrag 1 bis 6 eingetragen bei U( und O( |
| FKT | 0: O(<br>1: U( |
| BEF-REG | Befehlsregister |
| SAZ | Stepadresszähler |
| DB-ADR | Datenbausteinadresse |
| BST-STP | Bausteinstackpointer |
| NR | Bausteinnummer (OB, PB, FB, SB, DB) |
| REL-SAZ | relativer Stepadresszähler |

# Bedeutung der USTACK-Anzeigen

Wenn die CPU in den STOP-Zustand geht, ermitteln Sie mit folgender Tabelle die Fehlerursache.

| Fehlerbild | Störungsursache (Kennung im USTACK) | Beschreibung | Fehlerbeseitigung |
|---|---|---|---|
| Kein Neustart möglich | NINEU SYNFEH/ KOPFNI | Fehlerhafter Baustein:<br>• Komprimieren durch Netzausfall unterbrochen<br>• Bausteintransfer PG-AG durch Netzausfall unterbrochen<br>• Programmfehler (TIR/TNB/B MW) | URLÖSCHEN<br>Erneutes Laden des Programms |
| | KOLIF | DB 1 falsch programmiert | Überprüfen Sie<br>• die Kennungen für Koppelmerkerdefinitionen ("MASK01");<br>• die Kennung für den zu interpretierenden Teil des DB 1 ("DB1");<br>• die jeweiligen Endekennungen für Koppelmerkerdefinitionen bzw. für den zu interpretierenden Teil des DB 1 |
| | FEST | Fehler in der Selbsttestroutine der CPU | CPU tauschen |
| Fehlerhaftes Modul | ASPFA | Modulkennung nicht zulässig | zulässiges Modul einsetzen |
| Batterieausfall | BAU | Batterie nicht vorhanden oder entladen und Remanenz erwünscht | Batterie tauschen lassen (VIPA);<br>URLÖSCHEN<br>Programm neu laden |
| Peripherie unklar | PEU | Peripherie unklar | Stromversorgung prüfen |
| Unterbrechung der Programmbearbeitung | STOPS | Betriebsartenschalter auf STOP | Betriebsschalter auf RUN stellen |
| Unterbrechung der Programmbearbeitung | SUF | Substitutionsfehler: Funktionsbausteinaufruf mit fehlerhaftem Aktualparameter | FB-Aufruf korrigieren |

*Fortsetzung ...*

*... Fortsetzung*

| Fehlerbild | Störungsursache (Kennung im USTACK) | Beschreibung | Fehlerbeseitigung |
|---|---|---|---|
| | TRAF | Transferfehler:<br>• Programmierter DB-Befehl mit DW-Nummer > DB-Länge<br>• Programmierter DB-Befehl ohne vorherige DB-Eröffnung<br>• Zu erzeugender DB ist zu lang für den Anwenderspeicher (E DB-Operation) | Programmfehler beseitigen |
| | STS | • Software-Stop durch Anweisung (STP,STS)<br>• STOP-Anforderung vom PG | |
| | NNN | • Nicht dekodierbarer Befehl<br>• Parameterüberschreitung | Programmfehler beseitigen |
| | STUEB | Bausteinstacküberlauf:<br>• Es wurde die maximale Bausteinaufrufverschachtelung (32) überschritten<br>• Alarmgesteuertes Programm unterbricht das zyklische Programm während der Bearbeitung eines integrierten Hantierungsbausteins und im unterbrechenden alarmgesteuerten Programm wird ebenfalls ein integrierter Hantierungsbaustein aufgerufen. | Programmfehler beseitigen<br><br>Im zyklischen Programm Alarme sperren vor Aufruf der integrierten Hantierungsbausteine |
| | NAU | Netzausfall | |
| | QVZ | Quittungsverzug von der Peripherie:<br>• Es wurde im Programm ein nicht adressiertes Peripheriebyte angesprochen oder eine Peripheriebaugruppe quittiert nicht | |
| Unterbrechung der Programmbearbeitung | ZYK | Zykluszeitüberschreitung: Die Programmbearbeitung übersteigt die eingestellte Überwachungszeit | Programm auf Endlosschleifen überprüfen. Evt. Zykluszeit mit OB 31 nachtriggern oder Überwachungszeit verändern |

# Integrierte FBs und OBs

Das Systemprogramm der CPU 24x bietet Ihnen Sonderfunktionen, die Sie durch Aufruf von FBs oder OBs nutzen können. Diese Sonderfunktionen sind Bestandteil des Systemprogramms und belegen keinen Platz im Anwenderspeicher. Die Sonderfunktionen können zwar aufgerufen aber weder gelesen noch geändert werden.

Der Aufruf einer Sonderfunktion über FB oder OB gilt als Bausteinwechsel und beeinflusst die Schachtelungstiefe für Bausteine.

**Integrierte Sonder-FBs**

Die integrierten FBs sind Funktionsbausteine, die Sie benutzen können wie FBs, die im Anwenderspeicher liegen. Sie können die integrierten FBs weder lesen noch ändern.

Für die Offline-Programmierung bilden Sie in der aktuellen Programmdatei den Bausteinkopf des integrierten FBs nach. Hierzu programmieren Sie einen Leerbaustein, der die FB-Parameter und das Bausteinende BE enthält. Beim Aufruf sieht dann dieser Leerbaustein wie der Original-Baustein aus.

Die integrierten Sonderfunktionen können durch Alarme unterbrochen werden (keine Weckalarme).

Folgende Funktionsbausteine sind in die CPUs integriert.

| FB-Nr. | Kurzbeschreibung |
|---|---|
| FB 238 | Anwenderspeicher komprimieren |
| FB 239 | Anwenderbaustein löschen |
| FB 240 | Codewandler 4 Dekaden BCD-Festpunkt |
| FB 241 | Codewandler 16 Bit Festpunkt- BCD |
| FB 242 | Multipliziere 16 Bit Festpunkt |
| FB 243 | Dividiere 16 Bit Festpunkt |
| FB 244 | Daten senden |
| FB 245 | Daten empfangen |
| FB 246 | Daten holen |
| FB 247 | Ausarbeitung überwachen |
| FB 248 | Auftrag löschen |
| FB 249 | Schnittstelle einrichten |

**Integrierte Sonder-OBs**

Die integrierten OBs sind Sonderfunktionen, die Sie mit absoluten oder bedingten OB-Aufrufen nutzen können. Folgende Sonder-OBs sind zur Zeit integriert:

| OB-Nr. | Kurzbschreibung |
|---|---|
| OB 31 | Zykluszeit neu starten |
| OB 160 | Variable Wartezeit |
| OB 251 | PID-Algorithmus |

**DB 1**

Ein Default-DB 1 für die Parametrierung interner Funktionen.

**Hinweis!**

Nähere Beschreibung der FBs, OBs, und des DB 1 finden Sie im Kapitel "Integrierte Bausteine".

# Technische Daten

## CPU 24x

### Allgemein

| Elektrische Daten | VIPA 241-1BA01 ... VIPA 244-1BA01 |
|---|---|
| Spannungsversorgung<br>Stromaufnahme<br>Statusanzeigen (LEDs) | DC 24V<br>max. 1,5A<br>PW (grün) Versorgungsspannung<br>RN Run (grün): CPU befindet sich im RUN<br>STStop (rot): CPU befindet sich im STOP<br>QV Quittungsverzug (rot): Überschreitung der Quittungsverzugszeit<br>ZYZyklusverzug (rot): Überschreitung der Zyklusverzugszeit<br>BA BASP (rot): Befehlsausgabesperre ist aktiviert |
| Anschlüsse / Schnittstellen | 15 pol. Sub-D (Buchse) PG/OP-Buchse aktiv/passiv |
| Uhr, Pufferung | Lithium-Akku, 30 Tage Pufferung |
| Ausgangsstrom zum Rückwandbus | max. 3A |
| Mikroprozessor | 80C165 (16Bit) |
| Bearbeitungszeiten typ./k | 1,8ms |
| Kombination mit Peripheriemodulen | |
| max. Modulanzahl<br>max. digital E/A<br>max. analoge E/A | 32<br>32<br>16 |
| Maße und Gewicht | |
| Abmessungen (BxHxT) in mm<br>Gewicht | 25,4x76x76<br>80g |

### Modulspezifisch

| | CPU 241 | CPU 242 | CPU 243 | CPU 244 |
|---|---|---|---|---|
| Speicherplatz | 8 KByte | 32 KByte | 52 KByte | 104 KByte |
| Uhr | nein | nein | ja | ja |
| Merker/S-Merker | 1024/0 | 2048/0 | 2048/0 | 2048/8192 |
| Zeiten/Zähler | 32/32 | 64/64 | 128/128 | 128/128 |
| Adressierbare E/A digital/analog | 448/16 | 512/32 | 1024/256 | 1024/256 |
| Programm-bearbeitung | zyk. | zyk./Zeit | zyk./Zeit | zyk./Zeit/Alarm |
| Best.-Nr.: | VIPA 241-1BA01 | VIPA 242-1BA01 | VIPA 243-1BA01 | VIPA 244-1BA01 |

## CPU 24x NET

| Elektrische Daten | VIPA 241-2BT01/241-2BT10 ... VIPA244-2BT01/244-2BT10 |
|---|---|
| Spannungsversorgung | über Rückwandbus |
| Stromaufnahme | max. 380mA |
| Potentialtrennung | ≥ 500V AC |
| Statusanzeige (LEDs) | wie CPU 24x<br>zusätzlich mit<br>PW Versorgungsspannung CP<br>Tx, Rx Datentransfer-Anzeige |
| Anschlüsse/Schnittstellen | RJ45 für Twisted-Pair-Ethernet |
| Ethernet Schnittstelle | |
| Ankopplung | RJ45 |
| Netzwerk Topologie | Sterntopologie mit max. 2 Kopplern pro Segment |
| Medium | Twisted Pair |
| Übertragungsrate | 10 MBit / 10/100MBit (CPU 24x-2BT10) |
| Gesamtlänge | max. 100m pro Segment |
| Kombination mit Peripheriemodulen | |
| max. digitale E/A | 32 |
| max. analoge E/A | 16 |
| Maße und Gewicht | |
| Abmessungen (BxHxT) in mm | 50,8x76x76 |
| Gewicht | 80g |

## CPU 24x DP

| Elektrische Daten | VIPA 241-2BP01 ... VIPA 244-2BP01 |
|---|---|
| Spannungsversorgung | über Rückwandbus |
| Stromaufnahme | max. 380mA |
| Potentialtrennung | ≥ 500V AC |
| Statusanzeige (LEDs) | wie CPU 24x<br>zusätzlich mit<br>PW Power (grün): Versorgungsspannung 24V<br>ER Error (rot): Busausfall, Slave-Ausfall<br>RN Run (grün): Anzeige RUN-Betrieb-Zustands<br>DE Data exchange (gelb): Kommunikation mit dem Profibus.<br>IF Initialisierungsfehler (rot): Parametrierung falsch |
| Anschlüsse/Schnittstellen | 9-polig SubD (Buchse) Profibus-Ankopplung |
| **Profibus Schnittstelle** | |
| Ankopplung | 9-polige SubD-Buchse |
| Netzwerk Topologie | Linearer Bus, aktiver Busabschluss an beiden Enden, Stichleitungen sind möglich. |
| Medium | Abgeschirmtes verdrilltes Twisted Pair-Kabel, Schirmung darf, abhängig von Umgebungsbedingungen, entfallen. |
| Übertragungsrate | 9,6 kBaud bis 12 MBaud |
| Gesamtlänge | ohne Repeater 100m bei 12 MBaud mit Repeater bis 1000m |
| max. Teilnehmeranzahl | 32 Stationen in jedem Segment ohne Repeater. Mit Repeater erweiterbar auf 126. |
| **Kombination mit Peripheriemodulen** | |
| max. Anzahl Slaves | 125 |
| max. Anzahl Eingangs-Byte | 256 |
| max. Anzahl Ausgangs-Byte | 256 |
| **Maße und Gewicht** | |
| Abmessungen (BxHxT) in mm | 50,8x76x76 |
| Gewicht | 80g |

# Teil 3 Einsatz CPU 24x mit E/A Modulen

**Überblick**

In diesem Kapitel ist der Einsatz einer CPU 24x mit den Peripherie-Modulen des System 200V beschrieben.

Diese Angaben gelten auch für die CPU 24x NET und CPU 24x DP.

**Inhalt**


| Thema | Seite |
|---|---|
| **Teil 3 Einsatz CPU 24x mit E/A Modulen** | **3-1** |
| Einsatz in Verbindung mit E/A-Modulen | 3-2 |
| Automatische Vergabe von Peripherieadressen | 3-3 |
| Manuelle Vergabe von Peripherieadressen | 3-5 |
| Parametrierung von Peripheriemodulen | 3-6 |

**Hinweis!**

Diese Angaben gelten für alle in diesem Handbuch vorgestellten CPUs, da die Rückwandbus-Kommunikation zwischen CPU und Peripherie-Modulen immer gleich ist!

# Einsatz in Verbindung mit E/A-Modulen

**Allgemein**

Die hier gemachten Angaben beziehen sich immer auf Module, die auf dem gleichen Bus neben der CPU gesteckt sind.

Damit die gesteckten Peripheriemodule gezielt angesprochen werden können, müssen ihnen bestimmte Adressen in der CPU zugeordnet werden. Digitalbaugruppen werden im Allgemeinen byteweise, Analogbaugruppen wortweise angesprochen.

**Lese- bzw. Schreibzugriff über Peripheriebytes**

Beim Hochlauf der CPU werden die am Bus befindlichen Module ermittelt und ihnen Adressen im Peripheriebereich der CPU zugeordnet. Über Lese- bzw. Schreibzugriffe auf diese Peripheriebytes können Sie die Module ansprechen.

**Hinweis!**

Bitte beachten Sie, dass durch den lesenden und schreibenden Zugriff auf dieselbe Adresse unterschiedliche Module angesprochen werden.

Digitale und analoge Module haben getrennte Adressbereiche.

Digitalmodule: 0 ... 127

Analogmodule: 128 ... 255

**Adressbereiche über DB1 zuordnen**

Sie haben jederzeit die Möglichkeit über die DB1-Parametrierung die Adressbereiche neu zu belegen.

**Die Adresszuordnung und Parametrierung können Sie auch komfortabel unter WinNCS von VIPA durchführen!**

**Signalzustände in Prozessabbild**

Die Signalzustände der unteren Adresse (0 ... 127) werden zusätzlich in einem besonderen Speicherbereich, dem *Prozessabbild* gespeichert.

Das Prozessabbild ist in zwei Teile gegliedert:

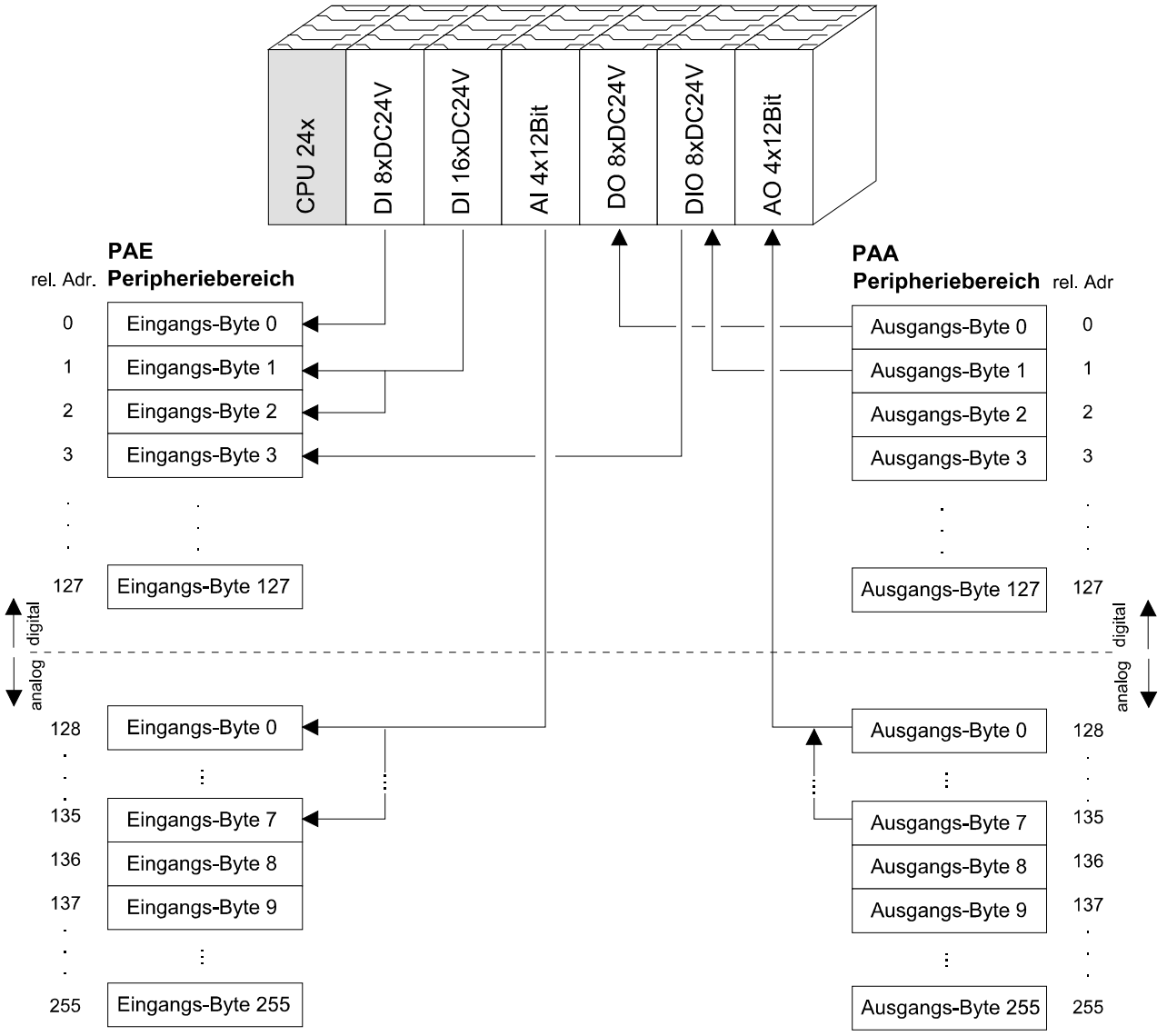
- Prozessabbild der Eingänge (PAE),
- Prozessabbild der Ausgänge (PAA).

# Automatische Vergabe von Peripherieadressen

Beim Hochlauf der CPU vergibt diese automatisch für die direkt gesteckten Module von 0 an aufsteigend Peripherieadressen für digitale Ein-/Ausgabe-Module.

Analog-Module werden auf gerade Adressen ab Adresse 128 abgelegt.

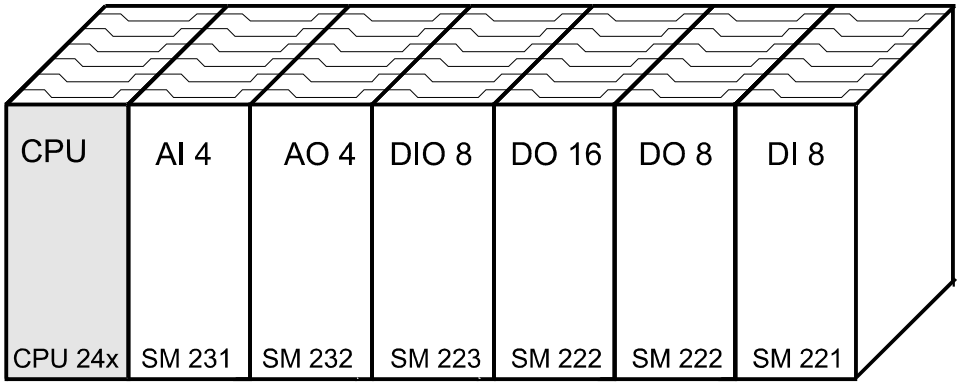
Die nachfolgende Abbildung soll die automatische Vergabe verdeutlichen:



**Beispiel**

Ihr System habe folgenden Aufbau:

*Adressierung*




Beim Hochlauf der CPU vergibt diese automatisch folgende Adressen für die direkt gesteckten Module sofern keine DB1-Parametrierung vorliegt:

| Modul<br>Speicherbedarf | AI 4<br>8Byte | AO 4<br>8Byte | DIO 8<br>je 1Byte | DO 16<br>2Byte | DO 8<br>1Byte | DI 8<br>1Byte |
|---|---|---|---|---|---|---|
| Eingabe-Adresse | 128 | | 0 | | | 1 |
| Ausgabe-Adresse | | 128 | 0 | 1 | 3 | |

Sie haben jederzeit die Möglichkeit über eine DB1-Parametrierung die Adressen manuell zu vergeben.

# Manuelle Vergabe von Peripherieadressen

Mit dem DB1 können Sie Adressen manuell vergeben. Bei der manuellen Adressvergabe über den DB1 sind folgende Regeln zu beachten:

- bei kombinierten Ein-/Ausgangs-Modulen ist zuerst die Eingangs- und dann die Ausgangsadresse zu parametrieren.
- Analog-Module können im Gegensatz zur automatischen Adressierung in den Peripheriebereich auf geraden Adressen ab 0 abgelegt werden. Sie liegen dann im Prozessabbild und erfahren in jedem Zyklus ein Update. Bitte beachten Sie, dass sich hierdurch die Zykluszeit erhöht.
- Jede Adresse besteht aus 2 Byte. Das High-Byte ist reserviert und muss immer 0 sein. In das Low-Byte tragen Sie die Modul-Adresse ein.

**Hinweis!**

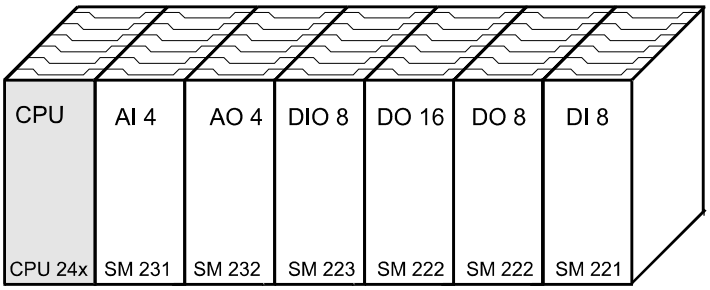
Sie können aber auch die Adressvergabe und Parametrierung unter **WinNCS** von VIPA durchführen. Hierbei wird ein DB1 erzeugt, den Sie als s5d-Datei exportieren und in Ihre CPU übertragen können.

**Beispiel**

Da beim Hochlauf der Modul-Typ erkannt wird, sind bei der DB1-Adressvergabe die neuen Adressen in der Reihenfolge anzugeben, in der die Module von links nach rechts neben der CPU gesteckt sind:

Das Beispiel soll die manuelle Vergabe der Adressen verdeutlichen:

Ihr System habe folgenden Aufbau:




Für diesen Aufbau möchten Sie z.B. folgende Adressen parametrieren:

| Modul<br>Speicherbedarf | AI 4<br>8Byte | AO 4<br>8Byte | DIO 8<br>je 1Byte | DO 16<br>2Byte | DO 8<br>1Byte | DI 8<br>1Byte |
|---|---|---|---|---|---|---|
| Eingabe-Adresse | 160 | | 24 | | | 25 |
| Ausgabe-Adresse | | 160 | 24 | 10 | 12 | |

Ergänzen Sie den DB1 um das Label UAT, gefolgt von der Adressierung im KC-Format:

```
KC= UAT: 0 160 0 160 0 24 0 24 0 10 0 12 0 25 ;
```

Beim Hochlauf der CPU wird der DB1 erkannt und die Adressen werden entsprechend zugeteilt.

# Parametrierung von Peripheriemodulen

Direkt gesteckte Module wie z.B. Analog-Module können bis zu 16 Byte-Parametrierdaten von der CPU erhalten. Im DB1 können Sie für jeden Steckplatz eine Parametrierung im Hex-Format angeben.

**Parametrierung abhängig vom CPU-Firmware-stand**

Jedem Steckplatz ist ein Label zugeordnet:

Steckplatz (0...31) → Label (P00...P31) CPU-Firmware-Stand bis V1.09

Steckplatz (1...32) → Label (P01...P32) CPU-Firmware-Stand größer V1.09

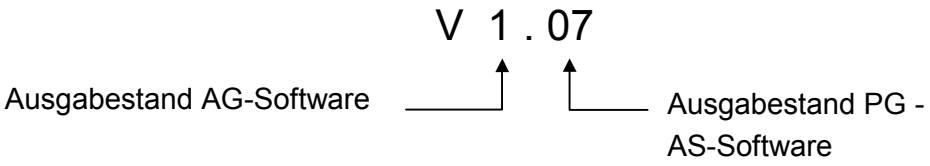
**Hinweis!**

Ab dem CPU-Firmware-Stand größer V1.09 beginnt die Steckplatznummerierung bei 1.

**Version bestimmen**

Die Version Ihrer CPU 24x können Sie den "Systeminformationen" Ihrer Programmiersoftware wie z.B. MC5 von VIPA entnehmen.

Sie setzt sich zusammen aus

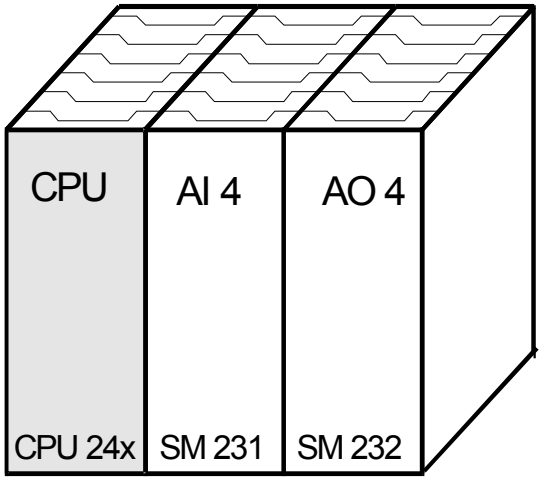


**Beispiel**

Das folgende Beispiel soll die Parametrierung anhand von zwei Analog-Modulen verdeutlichen:

**Beispiel**

Ihr System habe folgenden Aufbau:




**AI 4x16Bit Multiinput**
- kein Diagnosealarm
- Kanal 0: Spannungsmessung (-10 ... +10V)
- Kanal 1: Strommessung (4 ... 20mA)
- Kanal 2: Pt100 (-200 ... 850°C)
- Kanal 3: keine Messung
- Defaultoptionen

**AO 4x12Bit Multioutput**
- kein Diagnosealarm
- Kanal 0: Stromausgabe (4 ... 20mA)
- Kanal 1: Spannungsausgabe (1 ... 5V)
- Kanal 2: keine Ausgabe

Hieraus ergibt sich für die zwei Analog-Module folgende Parametrierung:

**Parameterbereiche :**

**AI 4x16Bit Multiinput**

| | | |
|---|---|---|
| Byte 0 | 00h | Diagnosealarm-Byte |
| Byte 1 | 00h | reserviert |
| Byte 2 | 2Bh | Funktions-Nr. Kanal 0 |
| Byte 3 | 2Dh | Funktions-Nr. Kanal 1 |
| Byte 4 | 02h | Funktions-Nr. Kanal 2 |
| Byte 5 | 00h | Funktions-Nr. Kanal 3 |
| Byte 6 | 00h | Option-Byte Kanal 0 |
| Byte 7 | 00h | Option-Byte Kanal 1 |
| Byte 8 | 00h | Option-Byte Kanal 2 |
| Byte 9 | 00h | Option-Byte Kanal 3 |

**AO 4x12Bit Multioutput**

| | | |
|---|---|---|
| Byte 0 | 00h | Diagnosealarm-Byte |
| Byte 1 | 00h | reserviert |
| Byte 2 | 04h | Funktions-Byte Kanal 0 |
| Byte 3 | 02h | Funktions-Byte Kanal 1 |
| Byte 4 | 00h | Funktions-Byte Kanal 2 |
| Byte 5 | 01h | Funktions-Byte Kanal 3 |

Ergänzen Sie im KC-Format (String) den DB1 um den Eintrag Pxx (xx für den Steckplatz) gefolgt von den Parameterbytes in Hex. Trennen Sie die Einträge durch ein Semikolon:

```
KC= P01: 00 00 2B 2D 02 00 00 00 00 00 ;
    P02: 00 00 04 02 00 01 ;
```

**Zwischen jedem Label, jeder Zahl und jedem Semikolon muss ein Leerzeichen stehen!**

# Teil 4 Einsatz CPU 24x-2BT10

**Überblick**

Im folgenden Kapitel ist der Einsatz der Net-CPU 24x-2BT10 und die Kommunikation unter TCP/IP beschrieben.

**Inhalt**


| Thema | Seite |
|---|---|
| **Teil 4 Einsatz CPU 24x-2BT10** | **4-1** |
| Industrial Ethernet in der Automatisierung | 4-2 |
| ISO/OSI-Schichtenmodell | 4-3 |
| Grundbegriffe | 4-6 |
| Protokolle | 4-7 |
| IP-Adresse und Subnetz | 4-10 |
| Planung eines Netzwerks | 4-12 |
| Kommunikationsmöglichkeiten des CP | 4-14 |
| Kommunikation programmieren | 4-16 |
| SEND/RECEIVE im SPS-Anwenderprogramm | 4-23 |
| Kopplung mit Fremdsystemen | 4-27 |
| NCM-Diagnose - Hilfe zur Fehlersuche | 4-30 |

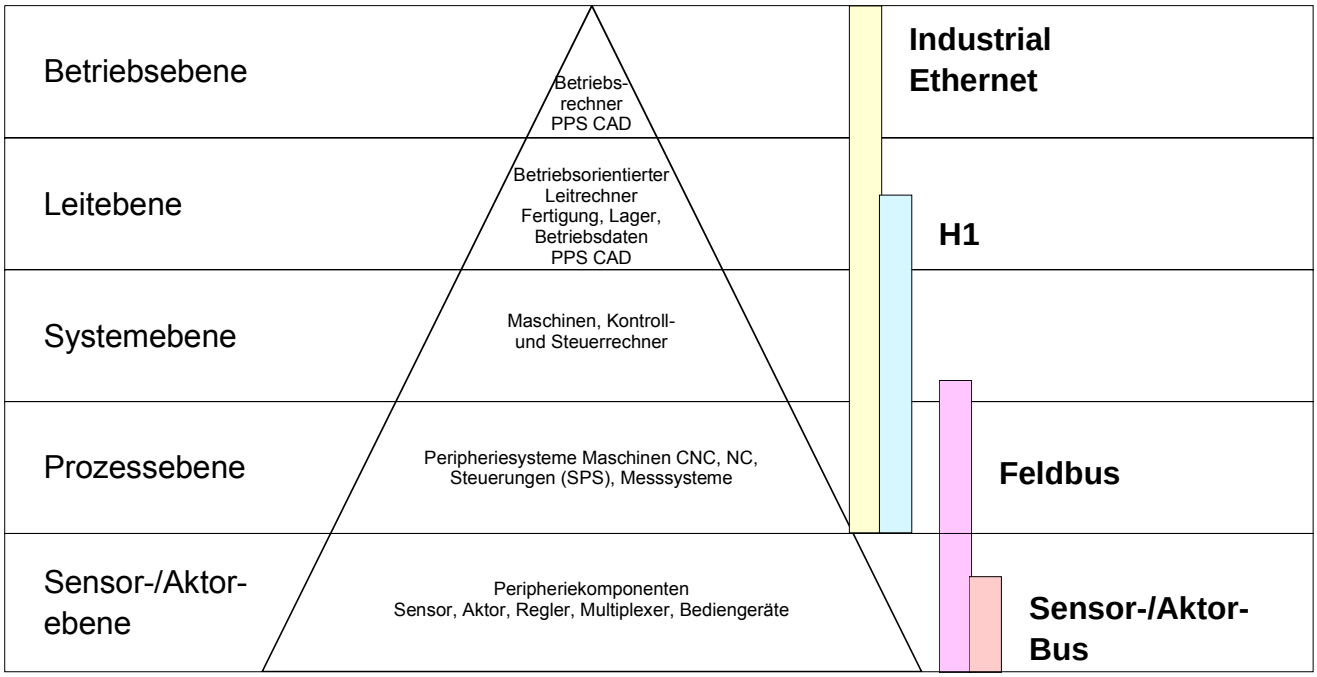
# Industrial Ethernet in der Automatisierung

**Übersicht**

Der Informationsfluss in einem Unternehmen stellt sehr unterschiedliche Anforderungen an die eingesetzten Kommunikationssysteme. Je nach Unternehmensbereich hat ein Bussystem unterschiedlich viele Teilnehmer, es sind unterschiedlich große Datenmengen zu übertragen, die Übertragungsintervalle variieren.

Aus diesem Grund greift man je nach Aufgabenstellung auf unterschiedliche Bussysteme zurück, die sich wiederum in verschiedene Klassen einteilen lassen.

Eine Zuordnung verschiedener Bussysteme zu den Hierarchieebenen eines Unternehmens zeigt das folgende Modell:




**Industrial Ethernet**

Physikalisch ist Industrial Ethernet ein elektrisches Netz auf Basis einer geschirmten Twisted Pair Verkabelung oder ein optisches Netz auf Basis eines Lichtwellenleiters.

Ethernet ist definiert durch den internationalen Standard IEEE 802.3. Der Netzzugriff bei Industrial Ethernet entspricht dem in der IEEE 802.3 festgelegten CSMA/CD-Verfahren (**C**arrier **S**ense **M**ultiple **A**ccess/**C**ollision **D**etection - Mithören bei Mehrfachzugriff/ Kollisionserkennung): Jeder Teilnehmer "hört" ständig die Busleitung ab und empfängt die an ihn adressierten Sendungen.

Ein Teilnehmer startet eine Sendung nur, wenn die Leitung frei ist. Starten zwei Teilnehmer gleichzeitig eine Sendung, so erkennen sie dies, stellen die Sendung ein und starten nach einer Zufallszeit erneut.

Durch Einsatz von Switches wird eine kollisionsfreie Kommunikation zwischen den Teilnehmern gewährleistet.

# ISO/OSI-Schichtenmodell

**Übersicht**

Das ISO/OSI-Schichtenmodell basiert auf einem Vorschlag, der von der International Standards Organization (ISO) entwickelt wurde. Es stellt den ersten Schritt zur internationalen Standardisierung der verschiedenen Protokolle dar. Das Modell trägt den Namen ISO-OSI-Schichtenmodell. OSI steht für **O**pen **S**ystem **I**nterconnection, die Kommunikation offener Systeme. Das ISO/OSI-Schichtenmodell ist keine Netzwerkarchitektur, da die genauen Dienste und Protokolle, die in jeder Schicht verwendet werden, nicht festgelegt sind. Sie finden in diesem Modell lediglich Informationen über die Aufgaben, die die jeweilige Schicht zu erfüllen hat.

Jedes offene Kommunikationssystem basiert heutzutage auf dem durch die Norm ISO 7498 beschriebenen ISO/OSI Referenzmodell. Das Referenzmodell strukturiert Kommunikationssysteme in insgesamt 7 Schichten, denen jeweils Teilaufgaben in der Kommunikation zugeordnet sind. Dadurch wird die Komplexität der Kommunikation auf verschiedene Ebenen verteilt und somit eine größere Übersichtlichkeit erreicht.

Folgende Schichten sind definiert:

| Schicht | Funktion |
|---|---|
| Schicht 7 | Application Layer (Anwendung) |
| Schicht 6 | Presentation Layer (Darstellung) |
| Schicht 5 | Session Layer (Sitzung) |
| Schicht 4 | Transport Layer (Transport) |
| Schicht 3 | Network Layer (Netzwerk) |
| Schicht 2 | Data Link Layer (Sicherung) |
| Schicht 1 | Physical Layer (Bitübertragung) |

Je nach Komplexität der geforderten Übertragungsmechanismen kann sich ein Kommunikationssystem auf bestimmte Teilschichten beschränken.

**Schichten**

**Schicht 1** Bitübertragungsschicht (physical layer)

Die Bitübertragungsschicht beschäftigt sich mit der Übertragung von Bits über einen Kommunikationskanal. Allgemein befasst sich diese Schicht mit den mechanischen, elektrischen und prozeduralen Schnittstellen und mit dem physikalischen Übertragungsmedium, das sich unterhalb der Bitübertragungsschicht befindet:

- Wie viel Volt entsprechen einer logischen 0 bzw. 1?
- Wie lange muss die Spannung für ein Bit anliegen?
- Pinbelegung der verwendeten Schnittstelle.

**Schicht 2** Sicherungsschicht (data link layer)

Diese Schicht hat die Aufgabe, die Übertragung von Bitstrings zwischen zwei Teilnehmern sicherzustellen. Dazu gehören die Erkennung und Behebung bzw. Weitermeldung von Übertragungsfehlern, sowie die Flusskontrolle.

Die Sicherungsschicht verwandelt die zu übertragenden Rohdaten in eine Datenreihe. Hier werden Rahmengrenzen beim Sender eingefügt und beim Empfänger erkannt. Dies wird dadurch erreicht, dass am Anfang und am Ende eines Rahmens spezielle Bitmuster gesetzt werden. In der Sicherungsschicht wird häufig noch eine Flussregelung und eine Fehlererkennung integriert.

Die Datensicherungsschicht ist in zwei Unterschichten geteilt, die LLC- und die MAC-Schicht.

Die MAC (**M**edia **A**ccess **C**ontrol) ist die untere Schicht und steuert die Art, wie Sender einen einzigen Übertragungskanal gemeinsam nutzen

Die LLC (**L**ogical **L**ink **C**ontrol) ist die obere Schicht und stellt die Verbindung für die Übertragung der Datenrahmen von einem Gerät zum anderen her.

**Schicht 3** Netzwerkschicht (network layer)

Die Netzwerkschicht wird auch Vermittlungsschicht genannt.

Die Aufgabe dieser Schicht besteht darin, den Austausch von Binärdaten zwischen nicht direkt miteinander verbundenen Stationen zu steuern. Sie ist für den Ablauf der logischen Verknüpfungen von Schicht 2-Verbindungen zuständig. Dabei unterstützt diese Schicht die Identifizierung der einzelnen Netzwerkadressen und den Auf- bzw. Abbau von logischen Verbindungskanälen. IP basiert auf Schicht 3.

Eine weitere Aufgabe der Schicht 3 besteht in der priorisierten Übertragung von Daten und die Fehlerbehandlung von Datenpaketen. IP (**I**nternet **P**rotokoll) basiert auf Schicht 3.

**Schicht 4** Transportschicht (transport layer)

Die Aufgabe der Transportschicht besteht darin, Netzwerkstrukturen mit den Strukturen der höheren Schichten zu verbinden, indem sie Nachrichten der höheren Schichten in Segmente unterteilt und an die Netzwerkschicht weiterleitet. Hierbei wandelt die Transportschicht die Transportadressen in Netzwerkadressen um.

Gebräuchliche Transportprotokolle sind: TCP, SPX, NWLink und NetBEUI.

**Schichten Fortsetzung ...**

**Schicht 5** Sitzungsschicht (session layer)

Die Sitzungsschicht wird auch Kommunikationssteuerungsschicht genannt. Sie erleichtert die Kommunikation zwischen Service-Anbieter und Requestor durch Aufbau und Erhaltung der Verbindung, wenn das Transportsystem kurzzeitig ausgefallen ist.

Auf dieser Ebene können logische Benutzer über mehrere Verbindungen gleichzeitig kommunizieren. Fällt das Transportsystem aus, so ist es die Aufgabe, gegebenenfalls eine neue Verbindung aufzubauen.

Darüber hinaus werden in dieser Schicht Methoden zur Steuerung und Synchronisation bereitgestellt.

**Schicht 6** Darstellungsschicht (presentation layer)

Auf dieser Ebene werden die Darstellungsformen der Nachrichten behandelt, da bei verschiedenen Netzsystemen unterschiedliche Darstellungsformen benutzt werden.

Die Aufgabe dieser Schicht besteht in der Konvertierung von Daten in ein beiderseitig akzeptiertes Format, damit diese auf den verschiedenen Systemen lesbar sind.

Hier werden auch Kompressions-/Dekompressions- und Verschlüsselungs-/Entschlüsselungsverfahren durchgeführt.

Man bezeichnet diese Schicht auch als Dolmetscherdienst. Eine typische Anwendung dieser Schicht ist die Terminalemulation.

**Schicht 7** Anwendungsschicht (application layer)

Die Anwendungsschicht stellt sich als Bindeglied zwischen der eigentlichen Benutzeranwendung und dem Netzwerk dar. Sowohl die Netzwerk-Services wie Datei-, Druck, Nachrichten-, Datenbank- und Anwendungs-Service als auch die zugehörigen Regeln gehören in den Aufgabenbereich dieser Schicht.

Diese Schicht setzt sich aus einer Reihe von Protokollen zusammen, die entsprechend den wachsenden Anforderungen der Benutzer ständig erweitert werden.

# Grundbegriffe

**Netzwerk (LAN)**

Ein Netzwerk bzw. LAN (**L**ocal **A**rea **N**etwork) verbindet verschiedene Netzwerkstationen so, dass diese miteinander kommunizieren können.

Netzwerkstationen können PCs, IPCs, TCP/IP-Baugrupen, etc. sein.

Die Netzwerkstationen sind, durch einen Mindestabstand getrennt, mit dem Netzwerkkabel verbunden. Die Netzwerkstationen und das Netzwerkkabel zusammen bilden ein Gesamtsegment. Alle Segmente eines Netzwerks bilden das Ethernet (Physik eines Netzwerks).

**Twisted Pair**

Früher gab es das Triaxial- (Yellow Cable) oder Thin Ethernet-Kabel (Cheapernet). Mittlerweile hat sich aber aufgrund der Störfestigkeit das Twisted Pair Netzwerkkabel durchgesetzt. Die CPU 24xNET hat einen Twisted Pair Anschluss.

Das Twisted Pair Kabel besteht aus 8 Adern, die paarweise miteinander verdrillt sind. Aufgrund der Verdrillung ist dieses System nicht so störanfällig wie frühere Koaxialnetze. Verwenden Sie für die Vernetzung Twisted Pair Kabel, die mindestens der Kategorie 5 entsprechen.

Abweichend von den beiden Ethernet-Koaxialnetzen, die auf einer Bus-Topologie aufbauen, bildet Twisted Pair ein Punkt-zu-Punkt-Kabelschema.

Das hiermit aufzubauende Netz stellt eine Stern-Topologie dar. Jede Station ist einzeln direkt mit dem Sternkoppler (Hub/Switch) zu einem Ethernet verbunden.

**Hub (Repeater)**

Ein Hub ist ein zentrales Element zur Realisierung von Ethernet auf Twisted Pair. Seine Aufgabe ist dabei, die Signale in beide Richtungen zu regenerieren und zu verstärken. Gleichzeitig muss er in der Lage sein, segmentübergreifende Kollisionen zu erkennen, zu verarbeiten und weiter zu geben. Er kann nicht im Sinne einer eigenen Netzwerkadresse angesprochen werden, da er von den angeschlossenen Stationen nicht registriert wird. Er bietet Möglichkeiten zum Anschluss an Ethernet oder zu einem anderen Hub bzw. Switch.

**Switch**

Ein Switch ist ebenfalls ein zentrales Element zur Realisierung von Ethernet auf Twisted Pair. Mehrere Stationen bzw. Hubs werden über einen Switch verbunden. Diese können dann, ohne das restliche Netzwerk zu belasten, über den Switch miteinander kommunizieren. Eine intelligente Hardware analysiert für jeden Port in einem Switch die eingehenden Telegramme und leitet diese kollisionsfrei direkt an die Zielstationen weiter, die am Switch angeschlossen sind. Ein Switch sorgt für die Optimierung der Bandbreite in jedem einzeln angeschlossenen Segment eines Netzes. Switches ermöglichen exklusiv nach Bedarf wechselnde Verbindungen zwischen angeschlossenen Segmenten eines Netzes.

# Protokolle

**Übersicht**

In Protokollen ist ein Satz an Vorschriften oder Standards definiert, der es Kommunikationssystemen ermöglicht, Verbindungen herzustellen und Informationen möglichst fehlerfrei auszutauschen. Ein allgemein anerkanntes Protokoll für die Standardisierung der kompletten Kommunikation stellt das ISO/OSI-Schichtenmodell dar (siehe "ISO/OSI-Schichtenmodell" weiter oben).

Folgende Protokolle kommen in der CPU 24xNET zum Einsatz:

- TCP/IP
- UDP
- RFC1006 (ISO-ON-TCP)

Nachfolgend sind diese Protokolle kurz aufgeführt:

**TCP/IP**

TCP/IP-Protokolle stehen auf allen derzeit bedeutenden Systemen zur Verfügung. Dies gilt am unteren Ende für einfache PCs, über die typischen Mini-Rechner, bis hinauf zu Großrechnern (auch für IBM-Systeme existieren TCP/IP-Implementierungen) und Spezialrechnern wie Vektorrechner und Parallelrechner.

Durch die weite Verbreitung von Internetzugängen und -Anschlüssen wird TCP/IP sehr häufig für den Aufbau heterogener Systemverbunde verwendet.

Hinter TCP/IP, das für die Abkürzungen **T**ransmission **C**ontrol **P**rotocol und **I**nternet **P**rotocol steht, verbirgt sich eine ganze Familie von Protokollen und Funktionen.

TCP und IP sind nur zwei der für den Aufbau einer vollständigen Architektur erforderlichen Protokolle. Die Anwendungsschicht stellt Programme wie "FTP" und "Telnet" auf PC-Seite zur Verfügung.

Die Anwendungsschicht des Ethernet-Teils der CPU 24xNET ist mit dem Anwenderprogramm unter Verwendung der Standardhantierungsbausteine definiert.

Diese Anwendungsprogramme nutzen für den Datenaustausch die Transportschicht mit den Protokollen TCP oder UDP, die wiederum mit dem IP-Protokoll der Internetschicht kommunizieren.

**IP**

Das IP (**I**nternet **P**rotokoll) deckt die Netzwerkschicht (Schicht 3) des ISO/OSI-Schichtmodells ab.

Die Aufgabe des IP besteht darin, Datenpakete von einem Rechner über mehrere Rechner hinweg zum Empfänger zu senden. Diese Datenpakete sind sogenannte Datagramme. Das IP gewährleistet weder die richtige Reihenfolge der Datagramme, noch die Ablieferung beim Empfänger.

Zur eindeutigen Unterscheidung zwischen Sender und Empfänger kommen 32Bit-Adressen (IP-Adressen) zum Einsatz, die bei *IPv4* in vier Oktetts (genau 8Bit) geschrieben werden, z.B. 172.16.192.11.

Diese Internetadressen werden weltweit eindeutig vergeben, so dass jeder Anwender von TCP/IP mit allen anderen TCP/IP Anwendern kommunizieren kann.

Ein Teil der Adresse spezifiziert das Netzwerk, der Rest dient zur Identifizierung der Rechner im Netzwerk. Die Grenze zwischen Netzwerkanteil und Host-Anteil ist fließend und hängt von der Größe des Netzwerkes ab.

Um IP-Adressen zu sparen, werden sogenannte *NAT-Router* eingesetzt, die eine einzige offizielle IP-Adresse besitzen und das Netzwerk hinter diesem Rechner abschotten. Somit können im privaten Netzwerk dann beliebige IP-Adressen vergeben werden.

**TCP**

Das TCP (**T**ransmission **C**ontrol **P**rotokoll) setzt direkt auf dem IP auf, somit deckt das TCP die Transportschicht (Schicht 4) auf dem OSI-Schichtenmodell ab. TCP ist ein verbindungsorientiertes End-to-End-Protokoll und dient zur logischen Verbindung zwischen zwei Partnern.

TCP gewährleistet eine folgerichtige und zuverlässige Datenübertragung. Hierzu ist ein relativ großer Protokoll-Overhead erforderlich, der folglich die Übertragung verlangsamt.

Jedes Datagramm wird mit einem mindestens 20 Byte langen Header versehen. In diesem Header befindet sich auch eine Folgenummer, mit der die richtige Reihenfolge erkannt wird. So können in einem Netzwerkverbund die einzelnen Datagramme auf unterschiedlichen Wegen zum Ziel gelangen.

Bei TCP-Verbindungen wird die Gesamtdatenlänge nicht übermittelt. Aus diesem Grund muss der Empfänger wissen, wie viele Bytes zu einer Nachricht gehören. Zur Übertragung von Daten mit variabler Länge können Sie die Längenangabe den Nutzdaten voranstellen und diese Längenangabe entsprechend auf der Gegenseite auswerten.

Eigenschaften

- Zur Adressierung werden neben der IP-Adresse Ports verwendet. Eine Port-Adresse sollte im Bereich 2000...65535 liegen. Ferne und lokale Ports dürfen bei nur 1 Verbindung identisch sein.
- Unabhängig vom eingesetzten Protokoll sind zur Datenübertragung auf SPS-Seite die Hantierungsbausteine FB 234 (SEND1) und FB 235 (RECV1) erforderlich.

**UDP**

Das UDP (**U**ser **D**atagramm **P**rotocol) ist ein verbindungsloses Transportprotokoll. Es wurde im RFC768 (Request for Comment) definiert. Im Vergleich zu TCP hat es wesentlich weniger Merkmale.

Die Adressierung erfolgt durch Portnummern.

UDP ist ein schnelles ungesichertes Protokoll, da es sich weder um fehlende Datenpakete kümmert, noch um die Reihenfolge der Pakete.

**ISO-on-TCP RFC1006**

Da der TCP-Transportdienst streamorientiert ist, bedeutet dies, dass einzelne vom Anwender zusammengestellte Datenpakete nicht unbedingt in der gleichen Paketierung beim Teilnehmer ankommen. Je nach Datenvolumen können Pakete zwar in der gleichen Reihenfolge aber anders paketiert ankommen, so dass der Empfänger die einzelnen Paketgrenzen nicht mehr erkennen kann. Beispielsweise werden 2x 10Byte-Pakete geschickt, die auf der Gegenseite als 20Byte-Paket ankommen. Aber gerade die richtige Paketierung ist für die meisten Anwendungen unerlässlich.

Dies bedeutet, dass oberhalb von TCP ein zusätzliches Protokoll erforderlich ist. Diese Aufgabe erfüllt der Protokollaufsatz RFC1006 (ISO-on-TCP). Der Protokollaufsatz beschreibt die Arbeitsweise einer ISO Transportschnittstelle (ISO 8072) auf der Basis des Transportinterfaces TCP (RFC793).

Das dem RFC1006 zugrunde liegende Protokoll ist in seinen wesentlichen Teilen identisch zu TP0 (Transport Protokoll, Class 0) in ISO 8073.

Da RFC1006 als Protokollaufsatz zu TCP gefahren wird, erfolgt die Dekodierung im Datenteil des TCP-Pakets.

**Eigenschaften**

- Im Gegensatz zu TCP wird hier der Empfang eines Telegramms bestätigt.
- Zur Adressierung werden neben der IP-Adresse anstelle von Ports TSAPs verwendet. Die TSAP-Länge kann 1 ... 16Byte betragen. Die Eingabe kann im ASCII- oder Hex-Format erfolgen. Ferne und lokale TSAPs dürfen bei nur 1 Verbindung identisch sein.
- Unabhängig vom eingesetzten Protokoll sind zur Datenübertragung auf SPS-Seite die Hantierungsbausteine FB 234 (SEND1) und FB 235 (RECV1) erforderlich.
- Im Gegensatz zu TCP können über RFC1006 unterschiedliche Telegrammlängen empfangen werden.

# IP-Adresse und Subnetz

**Aufbau IP-Adresse**

Industrial Ethernet unterstützt ausschließlich IPv4. Unter IPv4 ist die IP-Adresse eine 32-Bit-Adresse, die innerhalb des Netzes eindeutig sein muss und sich aus 4 Zahlen zusammensetzt, die jeweils durch einen Punkt getrennt sind. Jede IP-Adresse besteht aus einer **Net-ID** und **Host-ID** und hat folgenden Aufbau: **XXX.XXX.XXX.XXX**

Wertebereich: 000.000.000.000 bis 255.255.255.255

Die IP-Adressen werden vom Netzwerkadministrator vergeben.

**Net-ID**
**Host-ID**

Die **Net**work-ID kennzeichnet ein Netz bzw. einen Netzbetreiber, der das Netz administriert.

Über die Host-ID werden Netzverbindungen eines Teilnehmers (Hosts) zu diesem Netz gekennzeichnet.

**Subnet-Maske**

Die Host-ID kann mittels bitweiser UND-Verknüpfung mit der **Subnet-Maske** weiter aufgeteilt werden, in eine **Subnet-ID** und eine *neue* **Host-ID**.

Derjenige Bereich der ursprünglichen Host-ID, welcher von Einsen der Subnet-Maske überstrichen wird, wird zur Subnet-ID, der Rest ist die neue Host-ID.

| Subnet-Maske | binär alle "1" | | binär alle "0" |
|---|---|---|---|
| IPv4 Adresse | Net-ID | Host-ID | |
| Subnet-Maske und IPv4 Adresse | Net-ID | Subnet-ID | *neue* Host-ID |

**Subnetz**

Eine TCP-basierte Kommunikation per Punkt-zu-Punkt-, Hub- oder Switch-Verbindung ist nur zwischen Stationen mit identischer Network-ID und Subnet-ID möglich! Unterschiedliche Bereiche sind mit einem Router zu verknüpfen.

Über die Subnet-Maske haben Sie die Möglichkeit, die Ressourcen ihren Bedürfnissen entsprechend zu ordnen. So erhält z.B. jede Abteilung ein eigenes Subnetz und stört damit keine andere Abteilung.

**Adresse bei Erst-inbetriebnahme**

Bei der Erstinbetriebnahme besitzt der CP keine IP-Adresse. Die IP-Adresse können Sie mit dem FB 236 (IP_CONFIG) mittels eines Konfigurationsbausteins an den CP übergeben. Neben den IP-Adressdaten konfigurieren Sie hier auch Ihre Verbindungen.

**Adress-Klassen**

Für IPv4-Adressen gibt es fünf Adressformate (Klasse A bis Klasse E), die alle einheitlich 4 Byte = 32 Bit lang sind.

| | | | |
|---|---|---|---|
| Klasse A | 0 | Network-ID (1+7 bit) | Host-ID (24 bit) |
| Klasse B | 10 | Network-ID (2+14 bit) | Host-ID (16 bit) |
| Klasse C | 110 | Network-ID (3+21 bit) | Host-ID (8 bit) |
| Klasse D | 1110 | Multicast Gruppe | |
| Klasse E | 11110 | Reserved | |

Die Klassen A, B und C werden für Individualadressen genutzt, die Klasse D für Multicast-Adressen und die Klasse E ist für besondere Zwecke reserviert.

Die Adressformate der 3 Klassen A, B, C unterscheiden sich lediglich dadurch, dass Netzwork-ID und Host-ID verschieden lang sind.

Private IP Netze

Zur Bildung privater IP-Netze sind gemäß RFC1597/1918 folgende Adressbereiche vorgesehen:

| Netzwerk Klasse | von IP | bis IP | Standard Subnet-Maske |
|---|---|---|---|
| A | 10.0.0.0 | 10.255.255.255 | 255.0.0.0 |
| B | 172.16.0.0 | 172.31.255.255 | 255.255.0.0 |
| C | 192.168.0.0 | 192.168.255.255 | 255.255.255.0 |

(Die Host-ID ist jeweils unterstrichen.)

Diese Adressen können von mehreren Organisationen als Netz-ID gemeinsam benutzt werden, ohne dass Konflikte auftreten, da diese IP-Adressen weder im Internet vergeben noch ins Internet geroutet werden.

**Reservierte Host-IDs**

Einige Host-IDs sind für spezielle Zwecke reserviert.

| | |
|---|---|
| Host-ID = 0 | Identifier dieses Netzwerks, reserviert! |
| Host-ID = maximal (binär komplett Einsen) | Broadcast Adresse dieses Netzwerks |

**Hinweis!**

Wählen Sie niemals eine IP-Adresse mit Host-ID=0 oder Host-ID=maximal!

(z.B. ist für Klasse B mit Subnet-Maske = 255.255.0.0 die "172.16.0.0" reserviert und die "172.16.255.255" als lokale Broadcast-Adresse dieses Netzes belegt.)

# Planung eines Netzwerks

**Normen und Richtlinien**

Zur Kommunikation zwischen einzelnen Stationen gibt es gewisse Vorschriften und Regeln die einzuhalten sind. Hierbei werden die Form des Datenprotokolls, das Zugriffsverfahren auf den Bus und weitere, für die Kommunikation wichtige Grundlagen definiert.

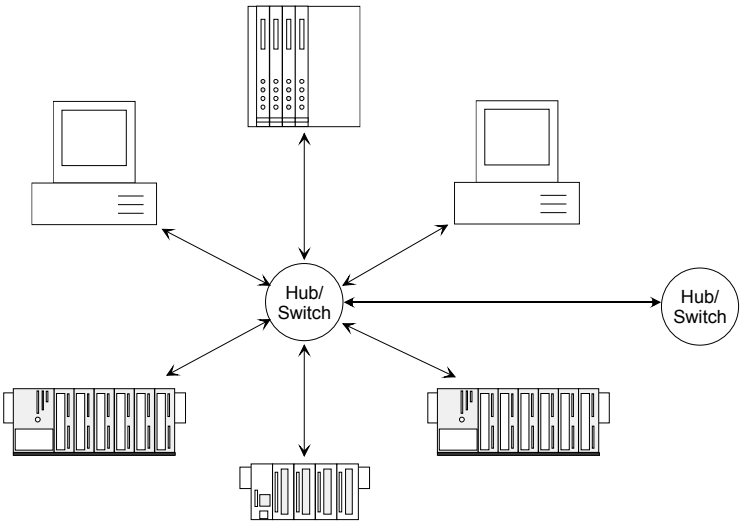
Basierend auf den von ISO festgelegten Standards und Normen wurde die CPU 24x-2BT10 von VIPA entwickelt.

In den folgenden internationalen und nationalen Gremien sind Normen und Richtlinien für Netzwerktechnologien festgelegt worden:

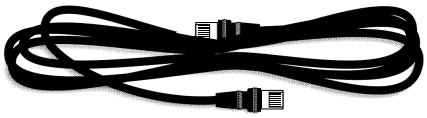
| | |
|---|---|
| ANSI | American National Standards Institute<br>Hier werden zur Zeit in der ANSI X3T9.5 Vereinbarungen für LANs mit hohen Übertragungsgeschwindigkeiten (100 MB/s) auf Glasfaserbasis formuliert. (FDDI) Fibre Distributed Data Interface. |
| CCITT | Committee Consultative Internationale de Telephone et Telegraph.<br>Von diesem beratenden Ausschuss werden unter anderem die Vereinbarungen für die Anbindung von Industriekommunikationsnetzen (MAP) und Büronetzen (TOP) an Wide Area Networks (WAN) erstellt. |
| ECMA | European Computer Manufacturers Association.<br>Hier werden verschiedene Standards für MAP und TOP erarbeitet. |
| EIA | Electrical Industries Association (USA)<br>Standardfestlegungen wie RS-232 (V.24) und RS-511 sind in diesem Ausschuss erarbeitet worden. |
| IEC | International Electrotechnical Commision.<br>Hier werden einzelne spezielle Standards festgelegt. z.B. für Feld Bus. |
| ISO | International Organisation for Standardization.<br>In diesem Verband der nationalen Normungsstellen wurde das OSI-Modell entwickelt (ISO/TC97/SC16). Es gibt den Rahmen vor, an den sich die Normungen für die Datenkommunikation halten sollen. ISO Standards gehen über in die einzelnen nationalen Standards wie z.B. UL und DIN. |
| IEEE | Institute of Electrical and Electronic Engineers (USA).<br>In der Projektgruppe 802 werden die LAN-Standards für Übertragungsraten von 1 bis 1000MB/s festgelegt. IEEE Standards bilden häufig die Grundlage für ISO-Standards z.B. IEEE 802.3 = ISO 8802.3. |

**Übersicht der Komponenten**

Der CP ist ausschließlich für den Einsatz in einem Twisted Pair Netz geeignet. Bei einem Twisted Pair Netz werden alle teilnehmenden Stationen sternförmig über Twisted Pair Kabel mit einem Hub/Switch verbunden, der seinerseits mit weiteren Hub/Switch kommunizieren kann. Zwei verbundene Stationen bilden ein Segment, wobei die Länge des Twisted Pair Kabels zwischen den Stationen max. 100m betragen darf.




Twisted Pair Kabel



Bei einem Twisted Pair Kabel handelt es sich um ein Kabel mit 8 Adern, die paarweise miteinander verdrillt sind.

Die einzelnen Adern haben einen jeweiligen Durchmesser von 0,4 bis 0,6mm. Verwenden Sie zur Vernetzung Twisted Pair Kabel, die mindestens der Kategorie 5 entsprechen.

**Ermitteln des Netzwerkbedarfs**

- Welche Fläche muss mit dem Kabelsystem abgedeckt werden?
- Wie viele Netzwerksegmente lösen am besten die physikalischen (räumlich, störungsbedingt) Gegebenheiten der Anlage?
- Wie viele Netzwerkstationen (SPS, IPC, PC, Transceiver, evtl. Bridges) sollen an das Kabelsystem angeschlossen werden?
- In welchem Abstand stehen die Netzwerkstationen voneinander getrennt?
- Welches "Wachstum" in Größe und Anzahl der Verbindungen muss das System bewältigen können?
- Welches Datenaufkommen ist zu bewältigen (Bandbreite, Zugriffe/Sec.)?

**Zeichnen des Netzwerkplans**

Zeichnen Sie Ihren Netzwerkplan. Bezeichnen Sie jedes Stück Hardware, das verwendet wird (wie Stationskabel, Hub, Switch). Halten Sie die Regeln und Grenzwerte im Auge.

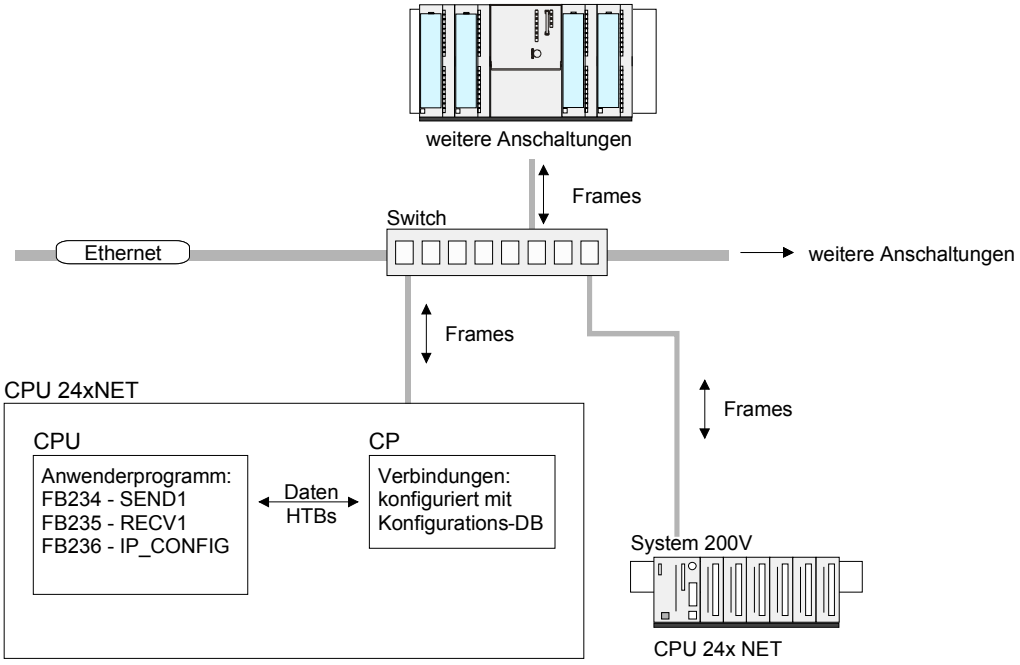
Messen Sie die Distanz zwischen allen Komponenten um sicher zu gehen, dass jeweils die maximale Länge nicht überschritten wird.

# Kommunikationsmöglichkeiten des CP

**Kommunikation zwischen CP 243 und CPU**

Der interne CP der CPU 24x-2BT10 ist über ein Dual-Port-RAM direkt mit der CPU verbunden. Auf CPU-Seite findet der Datenaustausch über die Hantierungsbausteine FB 234 (SEND1) und FB 235 (RECV1) statt.

Die Kommunikation über die entsprechenden Protokolle regeln Verbindungen, die mit dem FB 236 (IP_CONFIG) über den Konfigurations-Datenbaustein zu konfigurieren sind und über MMC oder die PG/OP-Schnittstelle in die CPU übertragen werden können.




**Kommunikationsarten**

Der CP unterstützt folgende Kommunikationsarten:

- Diagnose
- Projektierbare Verbindungen

**Diagnose**

Der CP unterstützt die NCM-Diagnose über Ethernet. Eine PG/OP-Kommunikation wird vom CP nicht unterstützt.

**Projektierbare Verbindungen**

Bei projektierbaren Verbindungen handelt es sich um Verbindungen zur Kommunikation zwischen SPS-Stationen. Die Verbindungen sind aus dem Anwenderprogramm über einen Konfigurations-Datenbaustein mit dem FB 236 (IP_CONFIG) zu konfigurieren.

## Funktons-übersicht

Nachfolgend sind die Funktionen aufgeführt die vom CP-Teil der CPU 24x-2BT10 unterstützt werden:

### Projektierbare Verbindungen

| Funktion | Eigenschaft |
|---|---|
| Maximale Anzahl Verbindungen | 64 |
| TCP-Verbindungen | SEND, RECEIVE, FETCH PASSIV, WRITE PASSIV<br>Verbindungsaufbau aktiv und passiv,<br>unterstützt unspezifizierten Verbindungspartner |
| ISO-on-TCP-Verbindungen<br>(RFC1006) | SEND, RECEIVE, FETCH PASSIV, WRITE PASSIV<br>Verbindungsaufbau aktiv und passiv,<br>unterstützt unspezifizierten Verbindungspartner |
| UDP-Verbindungen | SEND und RECEIVE<br>Die Übertragung der Telegramme erfolgt nicht quittiert, d.h. der Verlust von Nachrichten wird vom Sendebaustein nicht erkannt. |
| UDP-Broadcast-Verb. | SEND |
| UDP-Multicast-Verb. | SEND und RECEIVE (max. 16 Multicast-Kreise) |
| Datenblocklänge | max. 64kByte (max. 2kByte bei UDP) |
| VIPA-Hantierungs-bausteine | Für Verbindungsaufträge auf SPS-Seite:<br>FB 234 (SEND1) / FB 235 (RECV1) /<br>FB 236 (IP_CONFIG) |

### PG-Verbindungen und Diagnose

| Funktion | Eigenschaft |
|---|---|
| Maximale Anzahl PG/OP-Verbindungen | - |
| Diagnose | unterstützt NCM-Diagnose über Ethernet |
| Suche im Netzwerk | - |
| 10/100MBit | Umschaltung erfolgt automatisch |

# Kommunikation programmieren

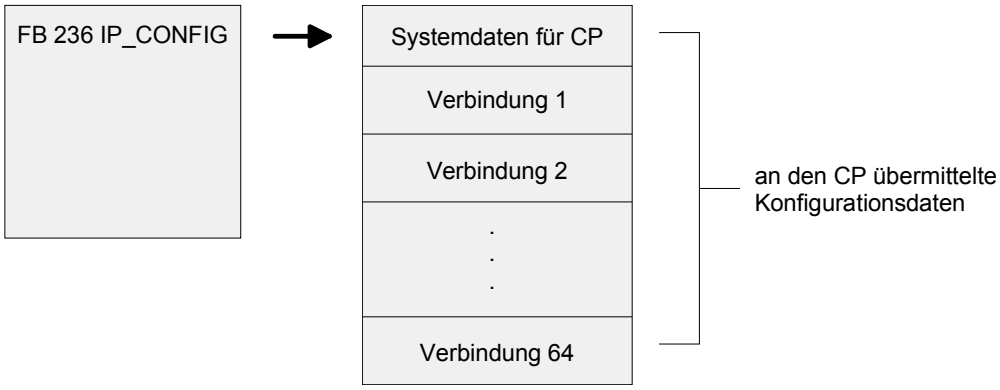
**Übersicht**

Die Konfiguration des CP erfolgt ausschließlich mit dem FB 236 (IP_CONFIG). Dieser ermöglicht Ihnen die flexible Übergabe von Datenbausteinen mit Projektierdaten an Ihren CP.

Für die Verarbeitung der Verbindungsaufträge auf SPS-Seite sind FB 234 (SEND1) und FB 235 (RECV1) vom Anwenderprogramm aufzurufen. Durch Einbindung dieser Bausteine in den Zyklus-OB können Sie zyklisch Daten senden und empfangen.

**Prinzip**

Über den FB 236 (IP_CONFIG), der im Anwenderprogramm aufgerufen wird, können Konfigurationsdaten für Kommunikationsverbindungen an den CP übertragen werden.




Der Konfigurations-DB kann jederzeit in den CP geladen werden.

**Achtung!**

Sobald das Anwenderprogramm über den FB 236 IP_CONFIG die Verbindungsdaten übergibt, schaltet die CPU den CP kurzzeitig in STOP. Der CP übernimmt die Systemdaten (inklusive IP-Adresse) und die neuen Verbindungsdaten und arbeitet diese im Anlauf ab (RUN).

## FB 236 - IP_CONFIG

Die Übertragung des Konfigurations-DB zum CP erfolgt mit einem einmaligen Aufruf des FB 236. Das Anwenderprogramm wird erst fortgesetzt, sobald der DB übertragen ist. Je nach Größe des Konfigurations-DB kann dies einige Zeit in Anspruch nehmen. Aus diesem Grund ist es sinnvoll, den FB 236 aus dem Anlauf-OB aufzurufen.

**Parameter**

| Parameter | Deklaration | Datentyp | Speicherbereich | Beschreibung |
|---|---|---|---|---|
| DBNR | INPUT | WORD | D | Geben Sie hier den Konfigurations-DB an, der die Systemdaten des CP und die Verbindungsparameter beinhaltet. |
| STAT | OUTPUT | WORD | E, A, M, D | Statusinformationen |

**Status-informationen**

| STATUS | Bedeutung |
|---|---|
| 0000h | Auftrag fertig ohne Fehler |
| 8181h | Auftrag läuft |
| 80B1h | Anzahl der zu sendenden Daten überschreitet die für diesen Dienst zulässige Obergrenze |
| 80C4h | Kommunikationsfehler<br>Fehler kann temporär auftreten, deshalb ist eine Wiederholung im Anwenderprogramm sinnvoll |
| 80D2h | Projektierungsfehler, die eingesetzte Baugruppe unterstützt diesen Dienst nicht. |
| 8183h | CP lehnt die angeforderte Datensatznummer ab. |
| 8184h | Systemfehler bzw. unzulässiger Parametertyp. |
| 8B01h | Kommunikationsfehler, DB konnte nicht übertragen werden. |
| 8B02h | Parameterfehler, Doppelter Parameterblock |
| 8B03h | Parameterfehler, Subblock im Parameterblock ist nicht erlaubt. |
| 8B04h | Parameterfehler, Länge die im FB angegeben wurde, stimmt nicht mit der Länge der Parameterblöcke/Subblöcke überein. |
| 8B05h | Parameterfehler, Länge des Parameterblocks ist ungültig. |
| 8B06h | Parameterfehler, Länge des Subblocks ist ungültig. |
| 8B07h | Parameterfehler, ID des Parameterblocks ist ungültig. |
| 8B08h | Parameterfehler, ID des Subblocks ist ungültig. |
| 8B09h | Systemfehler, Verbindung existiert nicht. |
| 8B0Ah | Datenfehler, Inhalt des Subblocks ist nicht korrekt. |
| 8B0Bh | Strukturfehler, Subblock ist doppelt aufgeführt. |

*Fortsetzung ...*

*... Fortsetzung*

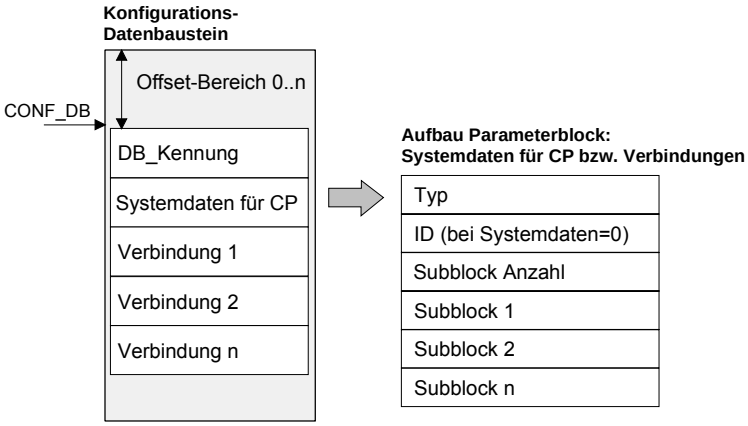
| STATUS | Bedeutung |
|---|---|
| 8B0Ch | Datenfehler, im Parameterblock sind nicht alle erforderlichen Parameter enthalten. |
| 8B0Eh | Datenfehler/Strukturfehler, Typ des CONF_DB ist ungültig. |
| 8B0Fh | Systemfehler, CP hat zu wenig Ressourcen, um den CONF_DB vollständig bearbeiten zu können. |
| 8B11 | Datenfehler, der angegebene Typ des Parameterblocks ist ungültig. |
| 8B12 | Datenfehler, es wurden zu viele Verbindungen angegeben. |
| 8F22h | Bereichslängenfehler beim Lesen eines Parameters. |
| 8F23h | Bereichslängenfehler beim Schreiben eines Parameters. |
| 8F24h | Bereichsfehler beim Lesen eines Parameters. |
| 8F25h | Bereichsfehler beim Schreiben eines Parameters. |
| 8F28h | Ausrichtungsfehler beim Lesen eines Parameters. |
| 8F29h | Ausrichtungsfehler beim Schreiben eines Parameters. |
| 8F32h | Parameter enthält eine zu große DB-Nummer. |
| 8F33h | DB-Nummernfehler |
| 8F7Fh | Interner Fehler |

**Konfigurations-Datenbaustein**

Der Konfigurations-Datenbaustein (CONF_DB) enthält sämtliche Verbindungsdaten und Konfigurationsdaten (IP-Adresse, Subnetz-Maske, Default-Router, NTP-Uhrzeit-Server und weitere) für einen Ethernet-CP. Der Konfigurations-DB wird mit dem FB 236 (IP_CONFIG) an den CP übergeben.

**Aufbau**

Der CONF_DB kann über einen Offset-Bereich an beliebiger Stelle innerhalb eines Datenbausteines beginnen. Jede Verbindung sowie die Systemdaten werden durch einen identisch aufgebauten Parameterblock beschrieben. Einzelne Parameter werden durch Subblöcke typisiert.




**Parameterblock für Systemdaten**

Nachfolgend finden Sie die für die Vernetzung des CPs relevanten Subblöcke. Diese sind im Parameterblock für Systemdaten anzugeben.
Je nach Anwendungsfall werden nicht alle Subblock-Typen benötigt.

Aufbau

| |
|---|
| **Typ = 0** |
| **ID = 0** |
| Subblock Anzahl = n |
| Subblock 1 |
| Subblock 2 |
| Subblock n |

| Subblock | | | | Parameter | |
|---|---|---|---|---|---|
| ID | Typ | Länge (Byte) | Bedeutung | Besonderheiten | Anwendung |
| 1 | SUB_IP_V4 | 4 + 4 | IP-Adresse gemäß IPv4 | Lokale IP-Adresse | zwingend |
| 2 | SUB_NETMASK | 4 + 4 | Subnetzmaske | - | zwingend |
| 4 | SUB_DNS_SERV_ ADDR | 4 + 4 | DNS Server Adresse | Kann bis zu 4 mal vorkommen. Erster Eintrag ist der Primary DNS Server. | optional |
| 8 | SUB_DEF_ROUTER | 4 + 4 | IP Adresse des Default Router | - | optional |
| 14 | SUB_DHCP_ENABLE | 1 + 4 | IP-Adresse von einem DHCP-Server beziehen | 0: kein DHCP<br>1: DHCP | optional |
| 15 | SUB_CLIENT_ID | Länge Client-ID + 4 | - | - | optional |

**Parameterblöcke für Verbindungstypen**

Sie erfahren nachfolgend, welche Werte in die Parameterblöcke einzutragen sind und welche Subblöcke zu den jeweiligen Verbindungstypen zu verwenden sind.
Je nach Anwendungsfall sind nicht alle Subblock-Typen erforderlich.
Von besonderer Bedeutung ist der ID-Parameter, der jedem Verbindungsparameterblock neben der Typkennung vorangestellt wird.
Bei programmierten Verbindungen können Sie diese ID innerhalb des zulässigen Wertebereichs frei vergeben. Sie müssen diese ID dann an der Aufrufschnittstelle der FBs für die SEND1/RECV1-Schnittstelle zur Identifizierung der Verbindung verwenden.

Wertebereich für die Verbindungs-ID: 1, 2 ... 64

TCP-Verbindung

| **Typ = 1** |
|---|
| **ID = Verbindungs-ID** |
| Subblock Anzahl = n |
| Subblock 1 |
| Subblock 2 |
| Subblock n |

| Subblock | | | | Parameter | |
|---|---|---|---|---|---|
| ID | Typ | Länge (Byte) | Bedeutung | Besonderheiten | Anwendung |
| 1 | SUB_IP_V4 | 4 + 4 | IP-Adresse gemäß IPv4 | IP-Adresse des Partners | zwingend*) |
| 9 | SUB_LOC_PORT | 2 + 4 | Lokaler Port | - | zwingend |
| 10 | SUB_REM_PORT | 2 + 4 | Ferner Port | - | zwingend*) |
| 18 | SUB_CONNECT_NAME | Länge Name + 4 | Name der Verbindung | - | optional |
| 19 | SUB_LOC_MODE | 1 + 4 | Lokale Betriebsart der Verbindung,Wertebereich:<br>0x00 = SEND/RECV<br>0x10 = S5-Adressierungsmodus bei FETCH/WRITE**)<br>0x80=FETCH**)<br>0x40=WRITE**)<br>Defaulteinstellung bei Verzicht auf den Parameter ist SEND/RECV.<br>FETCH/WRITE erfordern die Einstellung passiver Verbindungsaufbau. | - | optional |
| 21 | SUB_KBUS_ADR | - | - | Wert: fix 2 | optional |
| 22 | SUB_CON_ESTABL | 1 + 4 | Typ des Verbindungsaufbaus.<br>Legen Sie mit dieser Option fest, ob der Verbindungsaufbau von dieser Station aus erfolgen soll.<br>Wertebereich:<br>0 = passiv; 1 = aktiv | - | zwingend |

*) optional bei passiver Verbindung

**) die Codierungen können mittels ODER-Verknüpfungen kombiniert werden

UDP-Verbindung

| **Typ = 2** |
|---|
| **ID = Verbindungs-ID** |
| Subblock Anzahl = n |
| Subblock 1 |
| Subblock 2 |
| Subblock n |

| Subblock | | | | Parameter | |
|---|---|---|---|---|---|
| ID | Typ | Länge (Byte) | Bedeutung | Besonderheiten | Anwendung |
| 1 | SUB_IP_V4 | 4 + 4 | IP-Adresse gemäß IPv4 | IP-Adresse des Partners | zwingend |
| 9 | SUB_LOC_PORT | 2 + 4 | Lokaler Port | - | zwingend |
| 10 | SUB_REM_PORT | 2 + 4 | Ferner Port | - | zwingend |
| 18 | SUB_CONNECT_NAME | Länge Name + 4 | Name der Verbindung | - | optional |
| 19 | SUB_LOC_MODE | 1 + 4 | Lokale Betriebsart der Verbindung,Wertebereich:<br>0x00 = SEND/RECV<br>0x10 = S5-Adressierungsmodus bei FETCH/WRITE*)<br>0x80=FETCH*)<br>0x40=WRITE*)<br>Defaulteinstellung bei Verzicht auf den Parameter ist SEND/RECV.<br>FETCH/WRITE erfordern die Einstellung passiver Verbindungsaufbau. | - | optional |
| 21 | SUB_KBUS_ADR | - | - | Wert: fest auf 2 | optional |
| 23 | SUB_ADDR_IN_DATA_BLOCK | 1 + 4 | Freie UDP-Verbindung wählen.<br>Der ferne Teilnehmer wird vom Anwenderprogramm beim SEND1 Aufruf im Auftragsheader des Auftragspuffers eingetragen. Damit kann jeder beliebige Teilnehmer am Ethernet/LAN/WAN erreicht werden.<br>Wertebereich:<br>1 = Freie UDP-Verbindung<br>0 = sonst | Falls die Option "Freie UDP-Verbindung" gewählt wird, entfallen SUB_IP_V4, SUB_LOC_PORT, SUB_REM_PORT | optional |

*) die Codierungen können mittels ODER-Verknüpfungen kombiniert werden

ISO-on-TCP

| **Typ = 3** |
|---|
| **ID = Verbindungs-ID** |
| Subblock Anzahl = n |
| Subblock 1 |
| Subblock 2 |
| Subblock n |

| Subblock | | | | Parameter | |
|---|---|---|---|---|---|
| ID | Typ | Länge (Byte) | Bedeutung | Besonderheiten | Anwendung |
| 1 | SUB_IP_V4 | 4 + 4 | IP-Adresse gemäß IPv4 | IP-Adresse des Partners | zwingend*) |
| 11 | SUB_LOC_PORT | TSAP-Länge + 4 | Lokaler TSAP | - | zwingend |
| 12 | SUB_REM_PORT | TSAP-Länge + 4 | Ferner TSAP | - | zwingend*) |
| 18 | SUB_CONNECT_NAME | Länge Name + 4 | Name der Verbindung | - | optional |
| 19 | SUB_LOC_MODE | 1 + 4 | Lokale Betriebsart der Verbindung, Wertebereich:<br>0x00 = SEND/RECV<br>0x10 = S5-Adressierungsmodus bei FETCH/WRITE**)<br>0x80=FETCH**)<br>0x40=WRITE**)<br>Defaulteinstellung bei Verzicht auf den Parameter ist SEND/RECV.<br>FETCH/WRITE erfordern die Einstellung passiver Verbindungsaufbau. | - | optional |
| 21 | SUB_KBUS_ADR | - | - | Wert fest auf 2 | optional |
| 22 | SUB_CON_ESTABL | 1 + 4 | Typ des Verbindungsaufbaus.<br>Legen Sie mit dieser Option fest, ob der Verbindungsaufbau von dieser Station aus erfolgen soll.<br>Wertebereich:<br>0 = passiv; 1 = aktiv | - | zwingend |

*) optional bei passiver Verbindung

**) die Codierungen können mittels ODER-Verknüpfungen kombiniert werden

# SEND/RECEIVE im SPS-Anwenderprogramm

**Übersicht**

Für die Verarbeitung der Verbindungsaufträge auf SPS-Seite ist ein SPS-Anwenderprogramm in der CPU erforderlich. Hierbei kommen ausschließlich die Hantierungsbausteine FB 234 (SEND1) und FB 235 (RECV1) zum Einsatz. Beispielsweise durch Einbindung dieser Bausteine in den Zyklus-Baustein OB1 können Sie zyklisch Daten senden und empfangen.

**Kommunikations-bausteine**

Für die Kommunikation zwischen CPU und CP stehen Ihnen folgende FBs zur Verfügung:

FB 234 (SEND1)

Dieser Baustein übergibt die Nutzdaten aus dem über *TYPE*, *DBNR*, *OFFS* und *LEN* angegebenen Datenbereich an den über *ID* spezifizierten CP. Wurde der Datenbereich fehlerfrei übertragen, so wird "Auftrag fertig ohne Fehler" zurückgemeldet.

FB 235 (RECV1)

Der Baustein übernimmt vom CP die Nutzdaten und legt sie in dem über *TYPE*, *DBNR* und *OFFS* definieren Datenbereich ab. Wurde der Datenbereich fehlerfrei übernommen, so wird "Auftrag fertig ohne Fehler" zurückgemeldet.

Statusanzeigen

Der CP bearbeitet Sende- und Empfangsaufträge unabhängig vom CPU-Zyklus und benötigt hierzu eine Übertragungszeit. Die Schnittstelle mit den FB-Bausteinen zum Anwenderprogramm wird hierbei über Quittungen synchronisiert.

Für die Statusauswertung liefern die Kommunikationsbausteine Parameter zurück, die Sie in Ihrem Anwenderprogramm direkt auswerten können.

Diese Statusanzeigen werden bei jedem Baustein-Aufruf aktualisiert.

Einsatz unter hoher Kommunikationslast

Verwenden Sie keine zyklischen Aufrufe der Kommunikationsbausteine im OB1. Dies führt zu einer ständigen Kommunikation zwischen CPU und CP. Programmieren Sie stattdessen Ihre Kommunikationsbausteine in einem Zeit-OB, deren Zykluszeit größer ist als die des OB1 bzw. ereignisgesteuert.

Aufruf FB schneller als CP-Übertragungszeit

Wird ein Baustein im Anwenderprogramm erneut aufgerufen, bevor die Daten vollständig gesendet oder empfangen wurden, wird an der Schnittstelle der FBs wie folgt verfahren:

FB 234 (SEND1)

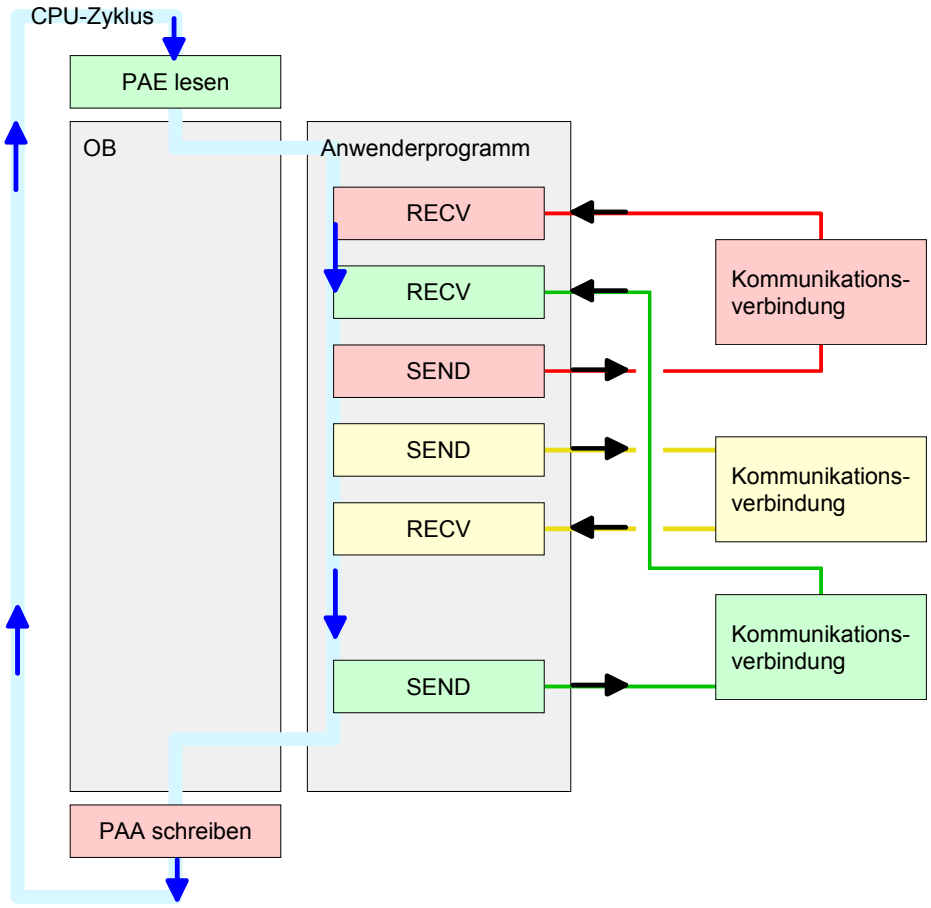
Es wird kein Auftrag entgegen genommen, bis die Datenübertragung über die Verbindung vom Partner quittiert wurde. Solange erhalten Sie die Meldung "Auftrag läuft", bis der CP den nächsten Auftrag für die gleiche Verbindung übernehmen kann.

FB 235 (RECV1)

Der Auftrag wird mit der Meldung "Es liegen noch keine Daten vor" quittiert, solange der CP die Empfangsdaten noch nicht vollständig empfangen hat.

**FB 234 (SEND1) u. FB 235 (RECV1) im Anwender-programm**

Eine mögliche Ablaufsequenz für die FBs zusammen mit den Organisations- und Programmbausteinen im CPU-Zyklus ist nachfolgend dargestellt:




Die FBs mit zugehöriger Kommunikationsverbindung sind farblich zusammengefasst. Hier können Sie auch erkennen, dass Ihr Anwender-programm aus beliebig vielen Bausteinen bestehen kann. Somit können Sie ereignis- bzw. programmgesteuert an beliebiger Stelle im CPU-Zyklus mit FB 234 (SEND1) Daten senden bzw. mit FB 235 (RECV1) Daten empfangen.

Sie können die Bausteine für **eine** Kommunikationsverbindung auch mehr-mals in einem Zyklus aufrufen.

**FB 234 (SEND1)** Mit FB 234 (SEND1) werden die zu sendenden Daten an den CP übertragen.

Parameter

| Parameter | Deklaration | Datentyp | Beschreibung |
|---|---|---|---|
| ACT | Input | WORD | Aktivierung des Senders<br>Wort = 0: Aktualisiert *DONE*, *ERR* und *STAT*<br>Wort = 1: Der unter DBNR mit Offset *OFFS* und der Länge *LEN* abgelegte Datenbereich wird gesendet. |
| ID | Input | WORD | Verbindungsnummer 1 ... 64 (identisch mit ID aus DB) |
| TYPE | Input | WORD | DB, MB, AB, EB<br>Sobald Sie als *Type* MB, EB oder AB angeben, ist der Parameter DBNR irrelevant. |
| DBNR | Input | WORD | Dieser Parameter ist nur erforderlich, wenn Sie als *Type* DB angeben. |
| OFFS | Input | WORD | Datenwort, ab dem die Daten zu senden sind |
| LEN | Input | WORD | Anzahl der Bytes, die aus dem DB zu senden sind |
| DONE | Output | WORD | Zustandsparameter für den Auftrag<br>Wort = 0: Auftrag läuft<br>Wort = 1: Auftrag fertig ohne Fehler |
| ERR | Output | WORD | Fehleranzeige<br>Wort = 0: Auftrag läuft (bei *DONE* = 0)<br>Wort = 0: Auftrag fertig ohne Fehler (bei *DONE* = 1)<br>Wort = 1: Auftrag fertig mit Fehler |
| STAT | Output | WORD | Statusanzeige, die in Verbindung mit *DONE* und *ERR* zurückgeliefert wird. Näheres hierzu finden Sie in der nachfolgenden Tabelle. |

**FB 235 (RECV1)** Mit FB 235 (RECV1) werden die Daten, die der CP empfangen hat, in die CPU übertragen.

Parameter

| Parameter | Deklaration | Datentyp | Beschreibung |
|---|---|---|---|
| ID | Input | WORD | Verbindungsnummer 1 ... 64 (Identisch mit ID aus DB) |
| TYPE | Input | WORD | DB, MB, AB, EB<br>Sobald Sie als *Type* MB, EB oder AB angeben, ist der Parameter DBNR irrelevant. |
| DBNR | Input | WORD | Dieser Parameter ist nur erforderlich, wenn Sie als *Type* DB angeben. |
| OFFS | Input | WORD | Datenwort, ab dem die empfangenen Daten abzulegen sind |
| NDR | Output | WORD | Zustandsparameter für den Auftrag<br>Wort = 0: Auftrag läuft<br>Wort = 1: Auftrag fertig ohne Fehler |
| ERR | Output | WORD | Fehleranzeige<br>Wort = 0: Auftrag läuft (bei *NDR* = 0)<br>Wort = 0: Auftrag fertig ohne Fehler (bei *NDR* = 1)<br>Wort = 1: Auftrag fertig mit Fehler |
| STAT | Output | WORD | Statusanzeige, die in Verbindung mit *NDR* und *ERR* zurückgeliefert wird. Näheres hierzu finden Sie in der nachfolgenden Tabelle. |
| LEN | Output | WORD | Anzahl der Bytes, die empfangen wurden |

**DONE, ERROR, STATUS**

In der nachfolgenden Tabelle sind alle Meldungen aufgeführt, die der CP nach einem SEND1-Auftrag bzw. RECV1-Auftrag zurückliefern kann.

Ein "-" bedeutet, dass diese Meldung für den entsprechenden SEND1- bzw. RECV1-Auftrag nicht existiert.

| DONE (SEND1) | NDR (RECV1) | ERR | STAT | Bedeutung |
|---|---|---|---|---|
| 1 | - | 0 | 0000h | Auftrag fertig ohne Fehler |
| - | 1 | 0 | 0000h | Neue Daten wurden ohne Fehler übernommen |
| 0 | - | 0 | 0000h | Kein Auftrag in Bearbeitung |
| - | 0 | 0 | 8180h | Es liegen noch keine Daten vor |
| 0 | 0 | 0 | 8181h | Auftrag läuft |
| 0 | 0 | 1 | 8183h | Für diesen Auftrag gibt es keine CP-Projektierung |
| 0 | - | 1 | 8184h | Es ist ein Systemfehler aufgetreten |
| - | 0 | 1 | 8184h | Es ist ein Systemfehler aufgetreten (Quelldatenbereich fehlerhaft) |
| 0 | - | 1 | 8185h | Parameter LEN größer als DB |
| | 0 | 1 | 8185h | Ziel-Puffer ist zu klein |
| 0 | 0 | 1 | 8186h | Parameter ID ungültig (nicht im Bereich 1 ... 64) |
| 0 | - | 1 | 8302h | keine Empfangsressourcen bei Ziel-Station, Empfänger-Station kann empfangene Daten nicht schnell genug verarbeiten bzw. hat keine Empfangsressourcen bereitgestellt. |
| 0 | - | 1 | 8304h | Die Verbindung ist nicht aufgebaut. Der Sendeauftrag sollte erst nach einer Wartezeit >100 ms erneut abgesetzt werden. |
| - | 0 | 1 | 8304h | Die Verbindung ist nicht aufgebaut. Der Empfangsauftrag sollte erst nach einer Wartezeit > 100ms erneut abgesetzt werden. |
| 0 | - | 1 | 8311h | Zielstation ist unter der angegebenen Ethernet-Adresse nicht erreichbar. |
| 0 | - | 1 | 8312h | Ethernet-Fehler im CP |
| 0 | | 1 | 8F22h | Quell-Bereich ungültig, wenn beispielsweise Bereich im DB nicht vorhanden Parameter LEN < 0 |
| - | 0 | 1 | 8F23h | Quell-Bereich ungültig, wenn beispielsweise Bereich im DB nicht vorhanden Parameter LEN < 0 |
| 0 | - | 1 | 8F24h | Bereichsfehler beim Lesen eines Parameters. |
| - | 0 | 1 | 8F25h | Bereichsfehler beim Schreiben eines Parameters. |
| 0 | - | 1 | 8F28h | Ausrichtungsfehler beim Lesen eines Parameters. |
| - | 0 | 1 | 8F29h | Ausrichtungsfehler beim Schreiben eines Parameters. |
| 0 | 0 | 1 | 8F32h | Parameter enthält zu große DB-Nummer. |
| 0 | 0 | 1 | 8F33h | DB-Nummer Fehler |
| 0 | 0 | 1 | 8F3Ah | Bereich nicht geladen (DB) |
| 0 | 0 | 1 | 8F7Fh | Interner Fehler z.B. Parameter LEN = 0 . |
| 0 | 0 | 1 | 80B1h | Die Längenangabe (im Parameter LEN) ist falsch. |
| 0 | 0 | 1 | 80C2h | Es liegt ein Auftragsstau vor. |
| 0 | 0 | 1 | 80C3h | Die Betriebsmittel (Speicher) der CPU sind temporär belegt. |
| 0 | 0 | 1 | 80C4h | Kommunikationsfehler (tritt temporär auf; daher ist eine Wiederholung im Anwenderprogramm sinnvoll.) |

Status-Parameter bei Neuanlauf

Bei einem Neuanlauf des CP werden die Ausgabe-Parameter wie folgt zurückgesetzt:

*DONE* = 0, *NDR* = 0, *ERR* = 0,
*STAT* = 8180h (RECV1), *STAT* = 8181h (SEND1)

# Kopplung mit Fremdsystemen

**Übersicht**

Die bei TCP- bzw. ISO-on_TCP unterstütze Betriebsart FETCH/WRITE können Sie prinzipiell für Zugriffe von Fremdgeräten auf den SPS-Systemspeicher verwenden. Damit Sie diesen Zugriff z.B. auch für PC-Anwendungen implementieren können, müssen Sie den Telegramm-Aufbau für die Aufträge kennen. Die spezifischen Header für Anforderungs- und Quittungstelegramme sind standardmäßig 16 Byte lang und werden auf den Folgeseiten beschrieben.

**ORG-Format**

Das Organisationsformat ist die Kurzbeschreibung einer Datenquelle bzw. eines Datenziels in SPS-Umgebung. Die verwendbaren ORG-Formate sind in der nachfolgenden Tabelle aufgelistet.

Die ERW-Kennung ist bei der Adressierung von Datenbausteinen relevant. In diesem Fall wird hier die Datenbaustein-Nummer eingetragen. Die Anfangsadresse und Anzahl adressieren den Speicherbereich und sind im HIGH-/LOW- Format abgelegt (Motorola - Adressformat).

| Beschreibung | Typ | Bereich |
|---|---|---|
| ORG-Kennung | BYTE | 1...x |
| ERW-Kennung | BYTE | 1...255 |
| Anfangsadresse | HILOWORD | 0...y |
| Länge | HILOWORD | 1...z |

In der nachfolgenden Tabelle sind die verwendbaren ORG-Formate aufgelistet. Die "Länge" darf nicht mit -1 (FFFFh) angegeben werden.

*ORG-Kennung 01h-04h*

| CPU-Bereich | DB | MB | EB | AB |
|---|---|---|---|---|
| ORG-Kennung | 01h | 02h | 03h | 04h |
| Beschreibung | Quell-/Zieldaten aus/in Datenbaustein im Hauptspeicher. | Quell-/Zieldaten aus/in Merkerbereich. | Quell-/Zieldaten aus/in Prozessabbild der Eingänge (PAE). | Quell-/Zieldaten aus/in Prozessabbild der Ausgänge (PAA). |
| ERW-Kennung (DBNR) | DB, aus dem die Quelldaten entnommen werden bzw. in den die Zieldaten transferiert werden. | irrelevant | irrelevant | irrelevant |
| Anfangsadresse Bedeutung | DBB-Nr., ab der die Daten entnommen bzw. eingeschrieben werden. | MB-Nr., ab der die Daten entnommen bzw. eingeschrieben werden. | EB-Nr., ab der die Daten entnommen bzw. eingeschrieben werden. | AB-Nr., ab der die Daten entnommen bzw. eingeschrieben werden. |
| Länge Bedeutung | Länge des Quell-/Zieldatenblocks in Worten. | Länge des Quell-/Zieldatenblocks in Bytes. | Länge des Quell-/Zieldatenblocks in Bytes. | Länge des Quell-/Zieldatenblocks in Bytes. |

**Hinweis!**

Informationen zu den erlaubten Bereichen finden Sie in den Technischen Daten ihrer CPU.

*ORG-Kennung 05h-07h*

| CPU-Bereich | PB | ZB | TB |
|---|---|---|---|
| ORG-Kennung | 05h | 06h | 07h |
| Beschreibung | Quell-/Zieldaten aus/in Peripheriebaugruppen. Bei Quelldaten Eingabebaugruppen, bei Zieldaten Ausgabebaugruppen. | Quell-/Zieldaten aus/in Zählerzellen. | Quell-/Zieldaten aus/in Zeitenzellen. |
| ERW-Kennung (DBNR) | irrelevant | irrelevant | irrelevant |
| Anfangsadresse Bedeutung | PB-Nr., ab der die Daten entnommen bzw. eingeschrieben werden. | ZB-Nr., ab der die Daten entnommen bzw. eingeschrieben werden. | TB-Nr., ab der die Daten entnommen bzw. eingeschrieben werden. |
| Länge Bedeutung | Länge des Quell-/Zieldatenblocks in Bytes. | Länge des Quell-/Zieldatenblocks in Worten (Zählerzelle = 1 Wort). | Länge des Quell-/Zieldatenblocks in Worten (Zählerzelle = 1 Wort). |

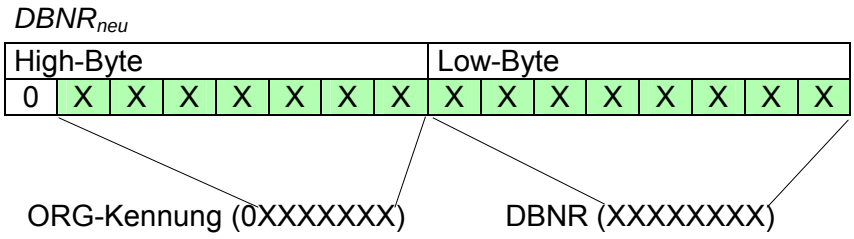
**Übertragen von Bausteinen mit Nummern >255**

*ORG-Kennung 81h-FFh*

Zur Übertragung von Datenbausteinen im Nummernbereich 256 ... 32768 können Sie die ORG-Kennung 81h-FFh verwenden.

Da die Angabe einer DB-Nr. >255 ein Wort als Länge erfordert, setzt sich $DBNR_{neu}$ aus dem Inhalt von ORG-Kennung und DBNR zusammen.

$DBNR_{neu}$ wird als Wort auf folgende Weise generiert:




Ist das höchste Bit der ORG-Kennung gesetzt, so ergibt sich das Low-Byte von $DBNR_{neu}$ aus der DBNR und das High-Byte von $DBNR_{neu}$ aus der ORG-Kennung, wobei das höchste Bit der ORG-Kennung eliminiert wird.

Folgende Formel soll dies nochmals verdeutlichen:

$DBNR_{neu}$=256 x (ORGKennung AND 7Fh) + DBNR

**Aufbau SPS-Header**

Bei FETCH und WRITE generiert der CP SPS-Header für Anforderungs- und Quittungstelegramme. Diese Header sind 16Byte lang und haben folgende Struktur:

**WRITE**

*Anforderungstelegramm Remote Station*

| | | |
|---|---|---|
| Systemkennung | ="S5" | (Wort) |
| Länge Header | =10h | (Byte) |
| Kenn. OP-Code | =01h | (Byte) |
| Länge OP-Code | =03h | (Byte) |
| **OP-Code** | **=03h** | (Byte) |
| ORG-Block | =03h | (Byte) |
| Länge ORG-Block | =08h | (Byte) |
| *ORG-Kennung** | | *(Byte)* |
| *ERW-Kennung* | | *(Byte)* |
| *Anfangsadresse* | | *(Wort)* |
| *Länge* | | *(Wort)* |
| Leerblock | =FFh | (Byte) |
| Länge Leerblock | =02h | (Byte) |
| Daten bis zu 64kByte (nur wenn Fehler-Nr.=0) | | |

*Quittungstelegramm CP*

| | | |
|---|---|---|
| Systemkennung | ="S5" | (Wort) |
| Länge Header | =10h | (Byte) |
| Kenn. OP-Code | =01h | (Byte) |
| Länge OP-Code | =03h | (Byte) |
| **OP-Code** | **=04h** | (Byte) |
| Quittungsblock | =0Fh | (Byte) |
| Länge Q-Block | =03h | (Byte) |
| Fehler-Nr. | | (Byte) |
| Leerblock | =FFh | (Byte) |
| Länge Leerblock | =07h | (Byte) |
| 5 leere Bytes angehängt | | |

**FETCH**

*Anforderungstelegramm Remote Station*

| | | |
|---|---|---|
| Systemkennung | ="S5" | (Wort) |
| Länge Header | =10h | (Byte) |
| Kenn. OP-Code | =01h | (Byte) |
| Länge OP-Code | =03h | (Byte) |
| **OP-Code** | **=05h** | (Byte) |
| ORG-Block | =03h | (Byte) |
| Länge ORG-Block | =08h | (Byte) |
| *ORG-Kennung** | | *(Byte)* |
| *ERW-Kennung* | | *(Byte)* |
| *Anfangsadresse* | | *(Wort)* |
| *Länge* | | *(Wort)* |
| Leerblock | =FFh | (Byte) |
| Länge Leerblock | =02h | (Byte) |

*Quittungstelegramm CP*

| | | |
|---|---|---|
| Systemkennung | ="S5" | (Wort) |
| Länge Header | =10h | (Byte) |
| Kenn. OP-Code | =01h | (Byte) |
| Länge OP-Code | =03h | (Byte) |
| **OP-Code** | **=06h** | (Byte) |
| Quittungsblock | =0Fh | (Byte) |
| Länge Q-Block | =03h | (Byte) |
| Fehler-Nr. | | (Byte) |
| Leerblock | =FFh | (Byte) |
| Länge Leerblock | =07h | (Byte) |
| 5 leere Bytes angehängt | | |
| Daten bis zu 64kByte (nur wenn Fehler-Nr.=0) | | |

*) Nähere Angaben zum Datenbereich finden Sie unter "ORG-Format" weiter oben.

**Meldungen von Fehler-Nr.**

Folgende Meldungen können über *Fehler-Nr.* zurückgeliefert werden:

| Fehler-Nr | Meldung |
|---|---|
| 00h | Kein Fehler aufgetreten |
| 01h | Der angegebene Bereich kann nicht gelesen bzw. beschrieben werden. |

# NCM-Diagnose - Hilfe zur Fehlersuche

**Checkliste zur Fehlersuche**

Diese Seite soll Ihnen bei der Fehlersuche dienen. Die nachfolgende Checkliste soll Ihnen helfen, einige typische Problemstellungen und deren mögliche Ursachen zu erkennen:

| Frage | Abhilfe bei "nein" |
|---|---|
| CPU im Run? | DC 24V-Spannungsversorgung überprüfen.<br>RUN/STOP-Schalter in Stellung RUN bringen.<br>SPS-Programm überprüfen und neu übertragen. |
| SEND1, RECV1 im Anwenderprogramm? | Für den Datentransfer zwischen CP und CPU sind diese 2 Bausteine im Anwenderprogramm erforderlich. Auch bei einer passiven Verbindung sind beide Bausteine aufzurufen |
| Kann CP verbinden? | Ethernetleitung überprüfen (bei Punkt-zu-Punkt-Verbindung ist ein gekreuztes Ethernetkabel zu verwenden).<br>IP-Adresse überprüfen. |
| Können Daten transferiert werden? | Port-Nr. für Lesen und Schreiben überprüfen.<br>Die Quell- und Zielbereiche überprüfen.<br>Prüfen, ob der richtige CP in der Wegewahl angewählt ist.<br>Den Empfangs- bzw. Sendepuffer vergrößern. |
| Wird der komplette Datenblock bei ISO-on-TCP gesendet? | Überprüfen Sie den LEN-Parameter bei SEND1.<br>Den Empfangs- bzw. Sendepuffer auf die erforderliche Größe einstellen. |

**Siemens NCM S7-Diagnose**

Der CP unterstützt das Siemens NCM-Diagnosetool. Das NCM-Diagnosetool ist Bestandteil des Siemens SIMATIC Managers. Dieses Tool liefert dynamisch Informationen zum Betriebszustand der Kommunikationsfunktionen von online geschalteten CPs.

Folgende Diagnose-Funktionen stehen Ihnen zur Verfügung:

- Betriebszustand an Ethernet ermitteln
- Im CP den Diagnosepuffer auslesen
- Verbindungen diagnostizieren

Auf den Folgeseiten finden Sie eine Kurzbeschreibung der NCM-Diagnose. Näheres zum Funktionsumfang und zum Einsatz des Siemens NCM-Diagnose-Tools finden Sie in der entsprechenden Online-Hilfe bzw. Dokumentation von Siemens.

**NCM-Diagnose starten**

Für den Aufruf des Diagnose-Tools haben Sie folgende 2 Möglichkeiten:

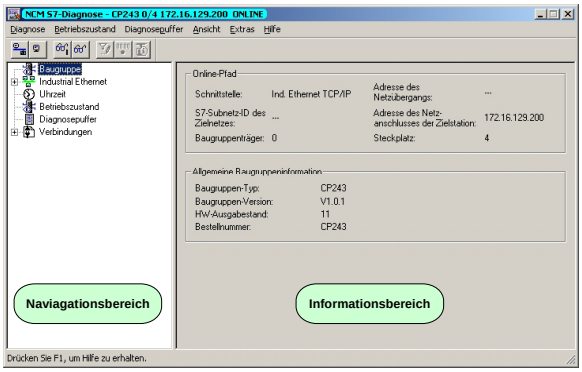
- Über *Windows-START-Menü > SIMATIC ... NCM S7 > Diagnose*
- Innerhalb der Projektierung bzw. Hardware-Konfiguration über das Register "Diagnose" im "Eigenschaften"-Dialog mit [Ausführen] die Diagnose aufrufen.

**Aufbau**

Die Arbeitsumgebung des Diagnose-Tools hat folgenden Aufbau:

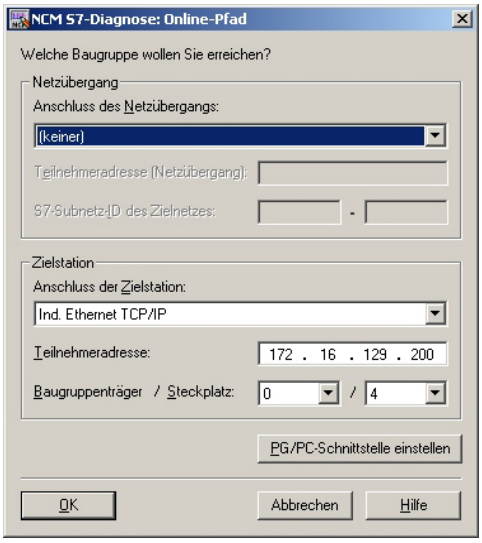
Im *Navigationsbereich* auf der linken Seite finden Sie die hierarchisch geordneten Diagnoseobjekte. Je nach CP-Typ und projektierter Verbindungen haben Sie eine angepasste Objektstruktur im Navigationsbereich.

Im *Inhaltsbereich* auf der rechten Seite finden Sie immer das Ergebnis der von Ihnen angewählten Navigationsfunktion im *Navigationsbereich*.




**Keine Diagnose ohne Verbindung**

Für eine Diagnose ist immer eine Online-Verbindung zu dem zu diagnostizierenden CP erforderlich. Klicken Sie hierzu in der Symbolleiste auf . Es öffnet sich folgendes Dialogfenster:




Stellen Sie unter *Zielstation* folgende Parameter ein:

**Anschluss...:** Ind. Ethernet TCP/IP

**Teilnehmer-Adr.:** Tragen Sie hier die IP-Adresse des CPs ein

**Baugruppenträger/Steckplatz:**

Für das System 200V ist immer *Baugruppenträger* 0 und *Steckplatz* 4 auszuwählen.

Stellen Sie Ihre PG/PC-Schnittstelle auf TCP/IP...RFC1006 ein. Mit [OK] starten Sie die Online-Diagnose.

**Diagnosepuffer auslesen**

Der CP besitzt einen Diagnosepuffer. Dieser hat die Architektur eines Ringspeichers. Hier können bis zu 100 Diagnosemeldungen festgehalten werden. In der NCM-Diagnose können Sie über das Diagnoseobjekt *Diagnosepuffer* die CP-Diagnosemeldungen anzeigen und auswerten.

Über einen Doppelklick auf eine Diagnosemeldung hält die NCM-Diagnose weitere Informationen bereit.

**Vorgehensweise bei der Diagnose**

Sie führen eine Diagnose aus, indem Sie ein Diagnoseobjekt im Navigationsbereich anklicken. Weitere Funktionen stehen Ihnen über das Menü und über die Symbolleiste zur Verfügung.

**Hinweis!**

Überprüfen Sie immer anhand der Checkliste am Kapitelanfang die Voraussetzungen für eine funktionsfähige Kommunikation.

Für den gezielten Diagnoseeinsatz ist folgende Vorgehensweise zweckmäßig:

- Diagnose aufrufen.
- Mit Dialog für Online-Verbindung öffnen, Verbindungsparameter eintragen und mit [OK] Online-Verbindung herstellen.
- Den CP identifizieren und über Baugruppenzustand den aktuellen Zustand des CPs ermitteln.
- Verbindungen überprüfen auf Besonderheiten wie:
  - Verbindungszustand
  - Empfangszustand
  - Sendezustand
- Über *Diagnosepuffer* den Diagnosepuffer des CP einsehen und entsprechend auswerten.
- Soweit erforderlich, Projektierung bzw. Programmierung ändern und Diagnose erneut starten.

# Teil 5 Einsatz CPU 24x-2BT01

**Überblick**

In folgendem Kapitel ist der Einsatz der Net-CPU 24x2BT01 NET und die Kommunikation unter H1 bzw. TCP/IP beschrieben. Sie finden hier auch eine Einführung in die Projektierung unter WinNCS mit einem konkreten Kommunikationsbeispiel.

**Inhalt**


| Thema | Seite |
|---|---|
| **Teil 5 Einsatz CPU 24x-2BT01** | **5-1** |
| Grundlagen | 5-2 |
| Planung eines Netzwerks | 5-7 |
| Standardisierung und Normen | 5-9 |
| Ethernet- und IP-Adresse | 5-10 |
| Projektierung der CPU 24x NET | 5-12 |
| Beispiel zur Projektierung | 5-16 |
| Anlaufverhalten | 5-29 |
| Systemeigenschaften der CPU 24x NET | 5-30 |
| Kopplung mit Fremdsystemen | 5-32 |
| Status- und Fehleranzeigen | 5-36 |
| Testprogramm für TCP/IP-Verbindungen | 5-44 |

# Grundlagen

**Netzwerk**

Ein Netzwerk verbindet verschiedene Netzwerkstationen so, dass diese miteinander kommunizieren können.

Netzwerkstationen können PCs, IPCs, H1/TCP/IP-Baugruppen, etc. sein.

Die Netzwerkstationen sind durch einen Mindestabstand getrennt mit dem Netzwerkkabel verbunden. Die Netzwerkstationen und das Netzwerkkabel zusammen bilden ein Gesamtsegment.

Alle Segmente eines Netzwerks bilden das Ethernet (Physik eines Netzwerks).

**Twisted Pair**

Früher gab es das Triaxial- (Yellow Cable) oder Thin Ethernet-Kabel (Cheapernet). Mittlerweile hat sich aber aufgrund der Störfestigkeit das Twisted Pair Netzwerkkabel durchgesetzt. Die CPU 24x NET hat einen Twisted-Pair-Anschluss.

Das Twisted Pair Kabel besteht aus 4 Adern, die paarweise miteinander verdrillt sind. Aufgrund der Verdrillung ist dieses System nicht so störanfällig wie ein Ethernet-System.

Abweichend von den beiden Ethernet-Koaxialnetzen, die auf einer Bus-Topologie aufbauen, bildet Twisted Pair ein Punkt-zu-Punkt-Kabelschema.

Das hiermit aufzubauende Netz stellt eine Stern-Topologie dar. Jede Station ist einzeln direkt mit dem Sternkoppler (Hub/Switch) zu einem Ethernet verbunden.

**Sternkoppler (Hub)**

Der Hub ist das zentrale Element zur Realisierung von Ethernet auf Twisted Pair.

Seine Aufgabe ist dabei, die Signale in beide Richtungen zu regenerieren und zu verstärken. Gleichzeitig muss er in der Lage sein, segmentübergreifende Kollisionen zu erkennen, zu verarbeiten und weiter zu geben. Er kann nicht im Sinne einer eigenen Netzwerkadresse angesprochen werden, da er von den angeschlossenen Stationen nicht registriert wird.

Er bietet Möglichkeiten zum Anschluss an Thin- bzw. Thick-Ethernet oder zu einem anderen Hub.

**Zugriffssteuerung**

Bei Ethernet gibt es das Prinzip des zufälligen Buszugriffs: Jeder Teilnehmer greift bei Bedarf von sich aus auf den Bus zu. Koordiniert wird der Buszugriff dabei durch das Verfahren CSMA/CD (Carrier Sense Multiple Access/Collision Detection - Mithören bei Mehrfachzugriff/ Kollisionserkennung): Jeder Teilnehmer "hört" ständig die Busleitung ab und empfängt die an ihn adressierten Sendungen.

Ein Teilnehmer startet eine Sendung nur, wenn die Leitung frei ist. Starten zwei Teilnehmer gleichzeitig eine Sendung, so erkennen sie dies, stellen die Sendung ein und starten nach einer Zufallszeit erneut.

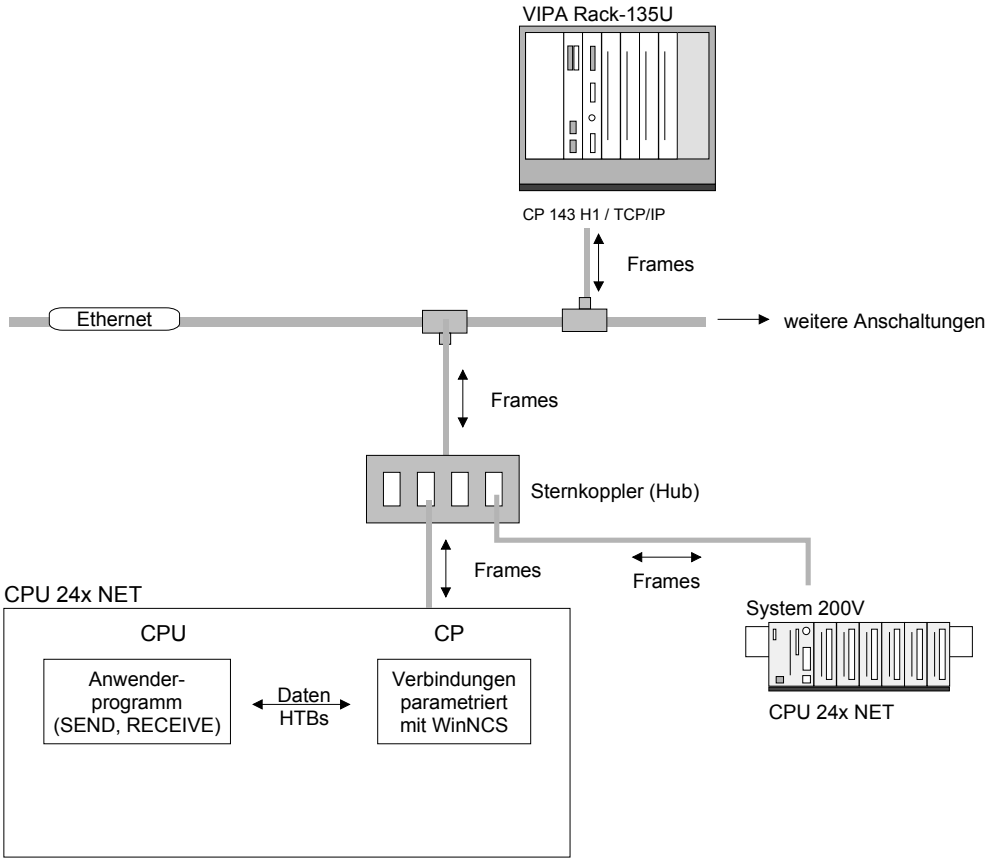
**Kommunikation**

Der interne CP der CPU 24x NET ist über ein Dual-Port-RAM direkt mit der CPU 24x verbunden. Das Dual-Port-RAM ist in 4 gleich große Segmente eingeteilt sog. "Kacheln".

Diese 4 Kacheln stehen auf der CPU-Seite als Standard-CP-Interface zur Verfügung. Der Datenaustausch findet über die Standard Hantierungsbausteine (SEND und RECEIVE) statt.

Die Kommunikation unter H1 bzw. TCP/IP regeln Verbindungen, die unter dem VIPA-Parametriertool WinNCS zu parametrieren sind und über das Netzwerk bzw. seriell in die CPU übertragen werden.

Näheres zur Parametrierung finden Sie im Handbuch von WinNCS (HB91).



**H1 - Industrial Ethernet**

H1 auch "Industrial Ethernet" genannt, ist ein Protokoll, das auf dem Ethernet-Standard aufsetzt.

Der Nachrichtenaustausch zwischen den Stationen erfolgt in H1 über H1-Frames, die über Transportverbindungen übertragen werden.

Eine Transportverbindung ist eine logische Verbindung zwischen zwei Zugangspunkten für Transport-Dienste auf verschiedenen Stationen. Eine Transportverbindung basiert auf Adressinformationen, die eindeutig den Transportweg zwischen den beiden Zugangspunkten beschreiben.

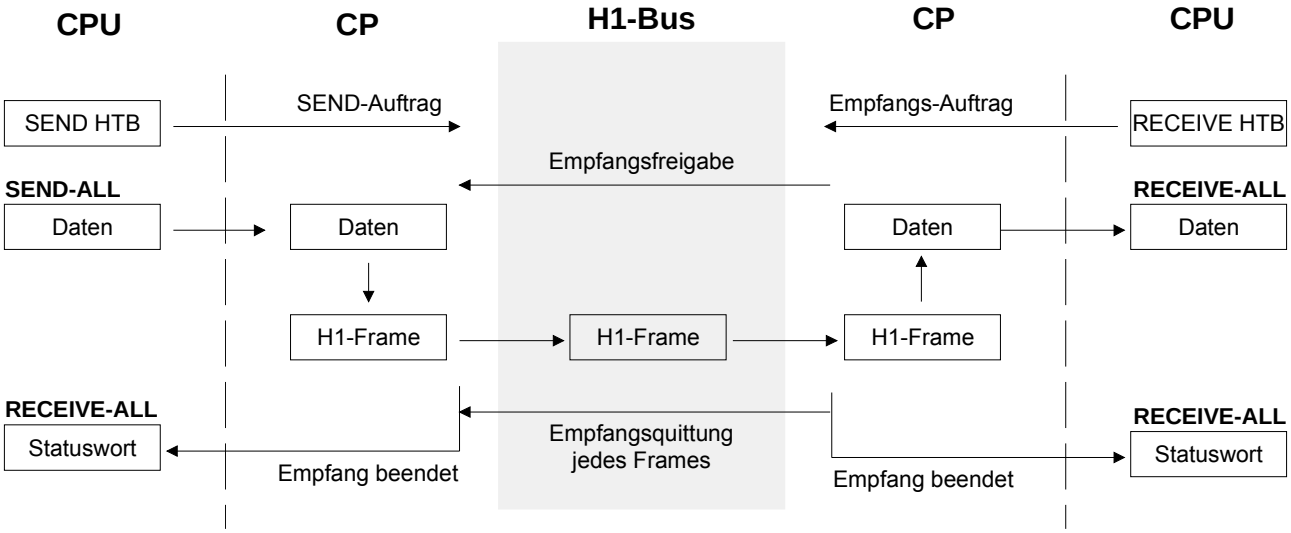
Eine Transportverbindung wird über folgende Parameter eindeutig beschrieben:

- *MAC-Adresse* auch Ethernetadresse, kennzeichnet eindeutig den Zugang zu einer Station.
- *TSAP* Transport-Service-Access-Points kennzeichnet Zugangskanäle für Dienste des Transportprotokolls.

Der CP stellt einen Datenpuffer bereit und transferiert über die Hintergrundkommunikation SEND-ALL die zu sendenden Daten in den Datenpuffer. Danach bildet der CP einen H1-Frame und überträgt, sobald eine Empfangsfreigabe erfolgt, dieses Frame an die Partnerstation. Nach dem Empfang des H1-Frames in der Partnerstation erhält der CP eine Empfangsquittung und transferiert über die Hintergrundkommunikation RECEIVE-ALL den Status des SEND-Auftrags in das zugehörige Anzeigenwort.

Auf diese Weise ist eine fehlerfreie Übertragung sichergestellt.

Die nachfolgende Abbildung soll dies nochmals verdeutlichen:




**Hinweis!**

Aufgrund der vielen Quittungstelegramme, die bei einer H1-Transport-Verbindung übertragen werden, ist die Netzbelastung unter H1 wesentlich höher als unter TCP/IP, wobei unter TCP/IP die Datensicherheit geringer ist!

**TCP/IP**

TCP/IP-Protokolle stehen auf allen derzeit bedeutenden Systemen zur Verfügung. Dies gilt am unteren Ende für einfache PCs, über die typischen Mini-Rechner, bis hinauf zu Großrechnern (auch für IBM-Systeme existieren TCP/IP-Implementierungen) und Spezialrechnern wie Vektorrechner und Parallelrechner. TCP/IP wird deshalb häufig für den Aufbau heterogener Systemverbunde verwendet.

Mit TCP/IP lassen sich umfassende, offene Netzwerklösungen zwischen den einzelnen Bereichen eines Unternehmens realisieren.

Mit TCP/IP lassen sich zum Beispiel folgende Aufgaben lösen:

- zentrale Steuerung und Überwachung von Fertigungsanlagen,
- Übermittlung von Maschinenzuständen,
- Managementinformationen,
- Produktionsstatistik,
- Übertragung großer Datenmengen.

TCP und IP sind nur zwei der für den Aufbau einer vollständigen Architektur erforderlichen Protokolle. Die Anwendungsschicht stellt Programme wie "FTP" und "Telnet" auf PC-Seite zur Verfügung.

Die Anwendungsschicht des CPs der CPU 24x NET ist mit dem Anwenderprogramm unter Verwendung der Standardhantierungsbausteine definiert.

Diese Anwendungsprogramme nutzen für den Datenaustausch die Transportschicht mit den Protokollen TCP oder UDP, die wiederum mit dem IP-Protokoll der Internetschicht kommunizieren.

*IP*

Die Hauptaufgabe von IP besteht in der Adressierung der Datenpakete. IP hat dieselbe Funktion wie ein Briefumschlag. Anhand der Adresse erkennt das Netzwerk den Bestimmungsort und kann die Datenpakete innerhalb des Netzwerks entsprechend weiterleiten.

Das Protokoll teilt die Daten in kleine Einheiten auf, da verschiedene Netzwerke unterschiedliche Datenlängen verwenden können.

Jedes Paket erhält eine Nummer, so dass der Empfang bestätigt werden kann und die Daten wieder rekonstruiert werden können. Um diese Folgenummern über das Netz senden zu können, hat TCP wie auch IP einen eigenen Umschlag, auf denen es die benötigten Nummern notiert.

*TCP*

Ein Paket mit Daten wird in einen TCP-Umschlag gepackt. Dieser wird seinerseits in einen IP-Umschlag gesteckt und an das Netzwerk gesendet. Hier sorgt TCP für den sicheren Transport der Daten durch das Netzwerk. TCP erkennt Übertragungsfehler und korrigiert diese.

Dieses Sicherheitssystem macht TCP-Verbindungen verhältnismäßig sicher.

UDP ist in diesem Fall wesentlich schneller. Es kümmert sich aber weder um fehlende Datenpakete, noch um die Reihenfolge der Pakete. UDP ist ein ungesichertes Protokoll.

**Dienste von TCP/IP**

OPEN / CONNECT

Eröffnung einer virtuellen Verbindung zu einem Partner im aktiven Fall, bzw. Warten auf die Verbindungsöffnung durch einen Partner im passiven Fall.

SEND

Übergabe eines Datenpuffers an TCP zum Versand an einen Partner.

RECEIVE

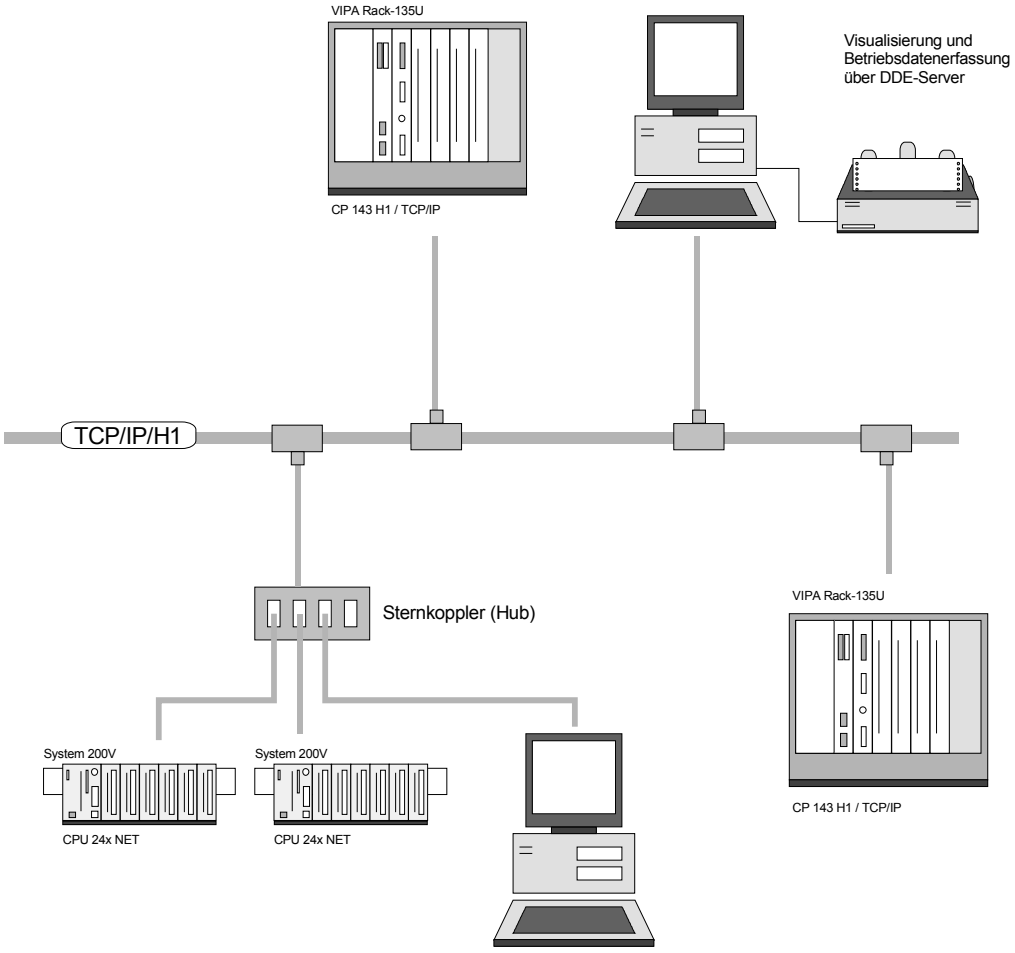
Empfang von Daten vom Partner.

CLOSE

Abbau einer virtuellen Verbindung.

**Anwendungs-beispiel für H1 bzw. TCP/IP**

*Einsatz unter TCP/P bzw. H1*

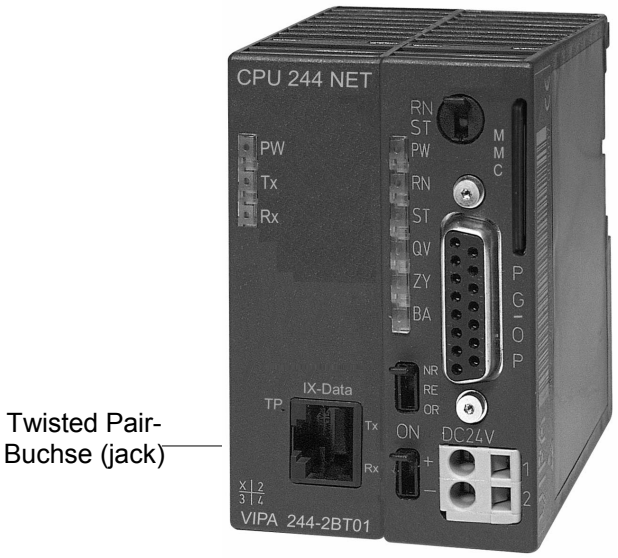


# Planung eines Netzwerks

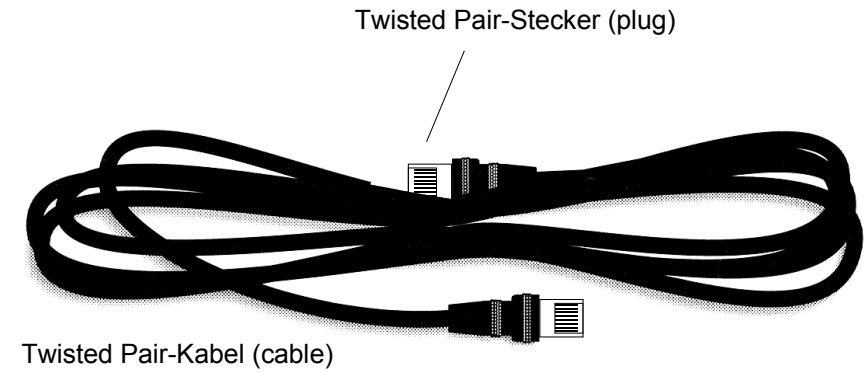
**Twisted Pair Netzwerk Hardware**

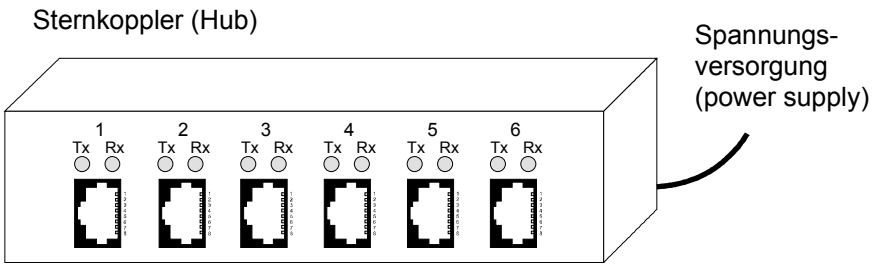
Sie können ein Twisted Pair-Netzwerk nur sternförmig aufbauen. Für die Sterntopologie ist ein Sternkoppler (Hub) erforderlich.

Bei einem Twisted Pair-Kabel handelt es sich um ein Kabel mit vier Adern, die paarweise miteinander verdrillt sind. Die einzelnen Adern haben einen jeweiligen Durchmesser von 0,4 bis 0,6 mm.




**CPU 244 NET**

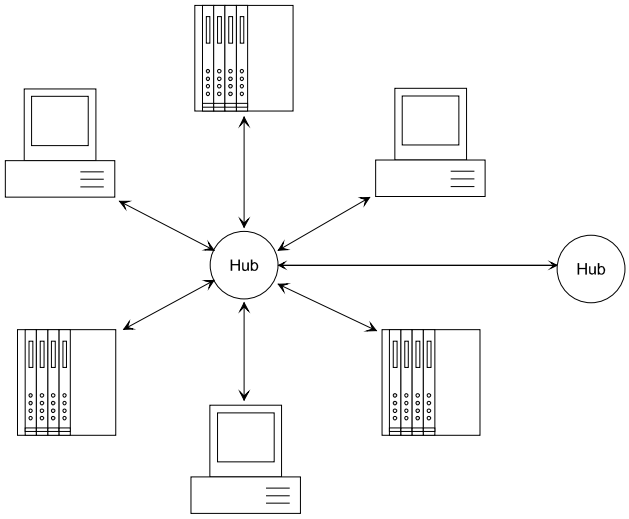


**Einschränkungen**

Hier ist eine Zusammenfassung der Einschränkungen und Regeln bezüglich Twisted Pair:

- Maximale Anzahl von Kopplerelementen pro Segment 2
- Maximale Länge eines Segments 100 m




**Ermitteln des Netzwerkbedarfs**

- Welche Fläche muss mit dem Kabelsystem abgedeckt werden?
- Wie viele Netzwerksegmente lösen am besten die physikalischen (räumlich, störungsbedingt) Gegebenheiten der Anlage?
- Wie viele Netzwerkstationen (SPS, IPC, PC, Transceiver, evtl. Bridges) sollen an das Kabelsystem angeschlossen werden?
- In welchem Abstand stehen die Netzwerkstationen voneinander getrennt?
- Welches "Wachstum" in Größe und Anzahl der Verbindungen muss das System bewältigen können?

**Zeichnen des Netzwerkplans**

Zeichnen Sie Ihren Netzwerkplan. Bezeichnen Sie jedes Stück Hardware das verwendet wird (wie Stationskabel, Hub). Halten Sie die Regeln und Grenzwerte im Auge.

Messen Sie die Distanz zwischen allen Komponenten um sicher zu gehen, dass die maximale Länge nicht überschritten wird.

# Standardisierung und Normen

Das Hauptkennzeichen der Busstruktur ist, dass nur ein einziger physikalischer Übertragungsweg existiert. Als physikalisches Übertragungsmedium wird dabei verwendet:

- ein oder mehrere elektrische Leitungen (verdrillte Leitung),
- Koaxialkabel (Triaxialkabel),
- Lichtwellenleiter.

Um die Kommunikation zwischen den einzelnen Stationen zu ermöglichen, müssen Vorschriften und Regeln verabredet und eingehalten werden.

Die Vereinbarungen regeln die Form des Datenprotokolls, das Zugriffsverfahren auf den Bus und weitere, für die Kommunikation wichtige Grundlagen.

Basierend auf den von ISO festgelegten Standards und Normen wurde die VIPA CPU 24x NET entwickelt.

**Normen und Richtlinien**

Folgende Normen und Richtlinien im Zusammenhang mit Netzwerktechnologien sind von internationalen und nationalen Gremien festgelegt worden:

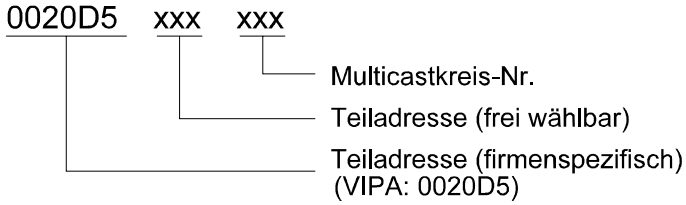
| | |
|---|---|
| ANSI | American National Standards Institute<br>Hier werden zur Zeit in der ANSI X3T9.5 Vereinbarungen für LANs mit hohen Übertragungsgeschwindigkeiten (100 MB/s) auf Glasfaserbasis formuliert. (FDDI) Fibre Distributed Data Interface. |
| CCITT | Committee Consultative Internationale de Telephone et Telegraph.<br>Von diesem beratenden Ausschuß werden unter anderem die Vereinbarungen für die Anbindung von Industriekommunikationsnetzen (MAP) und Büronetzen (TOP) an Wide Area Networks (WAN) erstellt. |
| ECMA | European Computer Manufacturers Association.<br>Hier werden verschiedene Standards für MAP und TOP erarbeitet. |
| EIA | Electrical Industries Association (USA)<br>Standardfestlegungen wie RS232 (V.24) und RS511 sind in diesem Ausschuß erarbeitet worden. |
| IEC | International Electrotechnical Commision.<br>Hier werden einzelne spezielle Standards festgelegt. z.B. für Feld Bus. |
| ISO | International Organisation for Standardization.<br>In diesem Verband der nationalen Normungsstellen wurde das OSI-Modell entwickelt (ISO/TC97/SC16). Es gibt den Rahmen vor, an den sich die Normungen für die Datenkommunikation halten sollen. ISO Standards gehen über in die einzelnen nationalen Standards wie z.B. UL und DIN. |
| IEEE | Institute of Electrical and Electronic Engineers (USA).<br>In der Projektgruppe 802 werden die LAN-Standards für Übertragungsraten von 1 bis 20 MB/s festgelegt. IEEE Standards bilden häufig die Grundlage für ISO-Standards z.B. IEEE 802.3 = ISO 8802.3. |

# Ethernet- und IP-Adresse

**Aufbau Ethernetadresse**

Mit der Ethernetadresse, auch MAC-Adresse genannt, wird eine Station adressiert. In einem Netz dürfen nicht mehrere Stationen mit der gleichen Ethernetadresse existieren.

Die Ethernetadresse hat folgenden Aufbau:




Die Ethernetadresse hat eine Länge von 6Byte. Die ersten 3 Byte legt der Hersteller fest. Diese 3Byte werden vom IEEE-Komitee vergeben. Die letzten 3Byte können frei vergeben werden.

In Anlagen werden die Ethernetadressen vom Netzwerkadministrator vergeben.

Die Broadcastadresse (Nachricht an alle Teilnehmer) ist immer:

FFFFFFFFFFFFh

**Aufbau IP-Adresse**

Die IP-Adresse ist eine 32-Bit-Adresse, die innerhalb des Netzes eindeutig sein muss. Die IP-Adresse setzt sich aus 4 Zahlen zusammen, die jeweils durch einen Punkt getrennt sind.

Die IP-Adresse hat folgenden Aufbau: **XXX.XXX.XXX.XXX**

Wertebereich: 000.000.000.000 bis 255.255.255.255

Auch die IP-Adressen werden vom Netzwerkadministrator vergeben.

Die Broadcastadresse (Nachricht an alle Teilnehmer) ist immer:

255.255.255.255

**Achtung!**

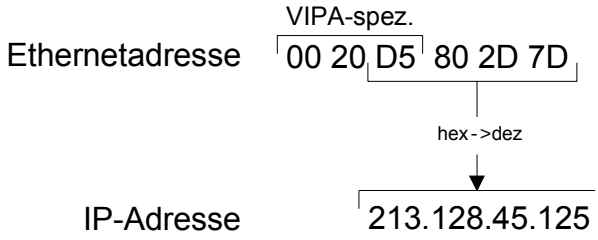
Es gibt IP-Adresse, die nicht verwendet werden dürfen! Diese Adressen sind für spezielle Dienste reserviert!

**Adresse bei Erstinbetriebnahme**

Bei der Erstinbetriebnahme einer CPU 24x NET besitzt die Baugruppe ihre ursprüngliche Ethernetadresse.

Sie finden diese Adresse auf einem Aufkleber, der sich an der Seite des Moduls befindet.

Aus dieser Ethernetadresse wird nur bei der Erstinbetriebnahme eine eindeutige IP-Adresse nach folgender Umrechnung gebildet.




**Hinweis!**

Eine Beziehung zwischen Ethernet- und IP-Adresse besteht nur bei der Erstinbetriebnahme.

Sie können jederzeit in WinNCS unter CP-Init eine andere Adresse zuweisen.

**Achtung!**

Die ursprüngliche Ethernetadresse kann nicht wiederhergestellt werden, da ein Urlöschen des CP-Teils nicht möglich ist.

# Projektierung der CPU 24x NET

**Übersicht**

Der CP-Teil verhält sich bei der Projektierung genauso wie die CP 143 H1/TCP/IP-Baugruppe von VIPA.

Die Projektierung besteht unter H1 und TCP/IP aus zwei Teilen:

- **CP-Projektierung** über WinNCS von VIPA (Verbindung zum Ethernet).
- **SPS-Programmierung** über Anwender-Programm (Verbindung zur SPS).

**CP-Projektierung mit WinNCS**

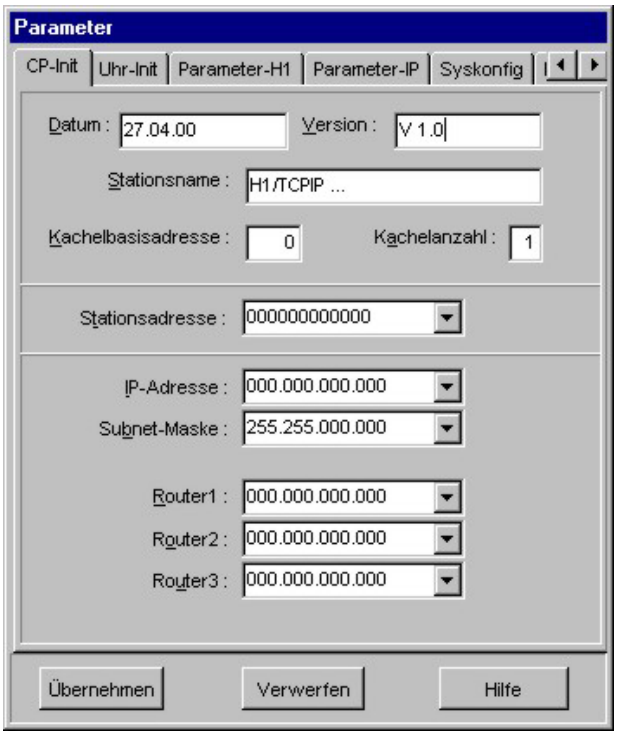
Die Projektierung des CP-Teils der CPU 24x NET findet ausschließlich unter WinNCS statt und gliedert sich in folgende 3 Teile:

- CP-Grundprojektierung,
- Verbindungsbausteine projektieren,
- Projektierdaten in den CP übertragen.

*CP-Grundprojektierung*

Hier werden Adresse und weitere Identifikationsparameter einer Station parametriert.

Fügen Sie unter der Funktionalität "Ethernet" eine neue Station im Netzwerkfenster ein und führen Sie im Parameterfenster die Parametrierung Ihrer Station durch.



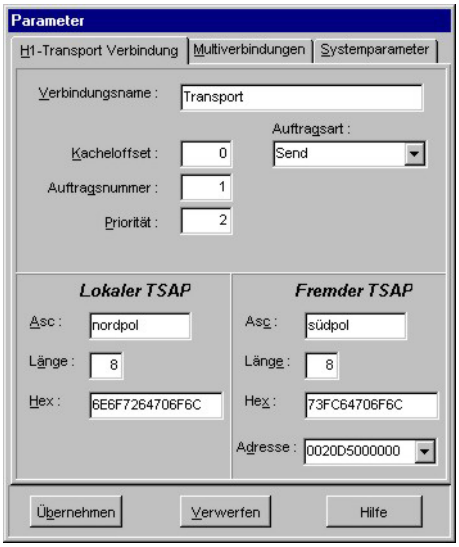

Über die CP-Grundprojektierung bestimmen Sie das Verhalten Ihrer Station im Netz.

*Verbindungsbausteine projektieren*

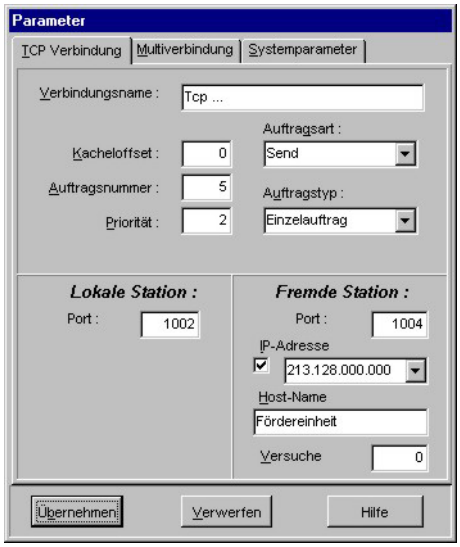
Ein Verbindungsbaustein beinhaltet die fernen, d.h. zum Partner im Netz orientierten und die lokalen, d.h. zum SPS-Programm orientierten Parameter einer Verbindung.

Abhängig von dem gewünschten Protokoll können Sie H1- bzw. TCP/IP-Verbindungen parametrieren, indem Sie bei angewähltem Stationssymbol die gewünschte Verbindung einfügen und parametrieren.

H1-Verbindung



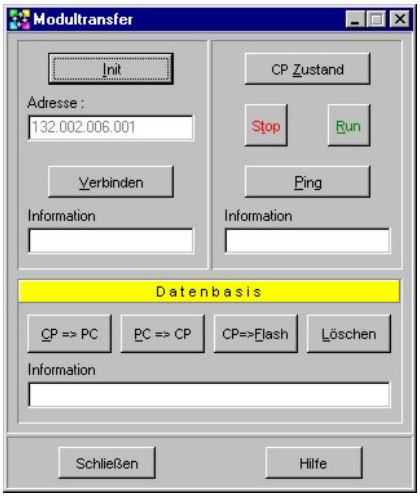

TCP/IP-Verbindung




*Projektierdaten in den CP übertragen*

Sind alle Verbindungen parametriert, müssen diese in den CP-Teil übertragen werden. Hierzu finden Sie unter WinNCS die "Modultransfer-Funktionen".

. Mit den Modultransferfunktionen können sie komfortabel über das Ethernet direkt den CP-Teil ansprechen und Ihre Daten in den CP-Teil übertragen.




Näheres hierzu finden Sie im Handbuch WinNCS HB91 in Kapitel "Ethernet-Funktionalität".

**Achtung!**

Bitte beachten Sie dass durch die Übertragung der Verbindungsparameter in den CP-Teil die Parameter in einem nicht gepufferten RAM abgelegt werden!

Nach der Übertragung sollten Sie in "Modultransfer" immer CP==>Flash ausführen. Ihre Daten werden dann in einem internen Flash-EPROM abgelegt!

**SPS-Anwender-Programmierung**

Für die Verarbeitung der Verbindungsaufträge auf SPS-Seite ist ein SPS-Anwenderprogramm in der CPU erforderlich. Hierbei kommen die Hantierungsbausteine (SEND, RECEIVE, ...) zum Einsatz, die u.a. in der CPU 24x NET integriert sind.

Auch bei der SPS-Programmierung ist zuerst ein Kommunikationskanal zwischen CPU und CP zu spezifizieren ("Synchronisation"). Hierzu dient der SYNCHRON-Baustein.

Sende- und Empfangsanstöße erfolgen über SEND bzw. RECEIVE. Und die Datenübertragung über SEND-ALL bzw. RECEIVE-ALL.

Eventuelle Fehlermeldungen erscheinen im Anzeigenwort.

Eine Beschreibung der Hantierungsbausteine finden Sie in diesem Handbuch im Teil "Integrierte Bausteine".

In der nachfolgenden Tabelle sind die erforderlichen Hantierungsbausteine aufgelistet.

| Hantierungsbaustein | | Erläuterung |
|---|---|---|
| SEND | FB244 | Senden eines Auftrags an den CP. |
| SEND-ALL | FB244 mit ANR=0 | Anstoß der Datenübernahme zwischen AG und CP. |
| RECEIVE | FB245 | Empfangen eines Auftrags vom CP. |
| RECEIVE-ALL | FB245 mit ANR=0 | Anstoß der Datenübernahme zwischen AG und CP. |
| FETCH | FB246 | Der FETCH-Baustein veranlasst das Holen von Daten. Ein Fetch-HTB ist ausschließlich mit der RW-Kennung zugelassen und liefert den Auftragsanstoß zum Lesen. |
| CONTROL | FB247 | Der CONTROL-Baustein wird für die Statusabfrage bzgl. eines Auftrags verwendet d.h. das ANZW eines definierten Auftrags wird aktualisiert. |
| RESET | FB248 | Der RESET-Baustein veranlasst das Rücksetzen eines Auftrags der angegebenen Verbindung. |
| RESET-ALL | FB245 mit ANR=0 | RESET-ALL erzwingt einen System-Reset des CPs. |
| SYNCHRON | FB249 | Der SYNCHRON stellt im Anlauf die Synchronisation zwischen AG und CP her. Gleichzeitig werden die Kachel gelöscht und die Blockgröße zwischen AG und CP ausgehandelt. Aktive Datenkommunikation kann nur über synchronisierte Kacheln stattfinden. |

### Synchronisation

Im Anlauf eines AGs muss jede parametrierte Schnittstelle eines CPs mit SYNCHRON synchronisiert werden. Dies gilt für jede Anlaufart eines AGs:

- OB21 für manuellen Wiederanlauf,
- OB22 für Wiederanlauf nach Netzspannungsausfall.

**OB22 bzw. OB21**

```
00000      :
00002      :SPA FB 22           Wartezeit und synchron
      NAME #ANLAUF
00009      :BE
```

**FB22**

```
      BSTNAME #NETZEIN
00022      :SPA FB 249          Synchron-Baustein
      NAME #SYNCHRON
      SSNR =KY 0,0             SSNR bzw. Kachelbasis 0
      BLGR =KY 0,6             Blockgrösse 6 (512 Byte)
      PAFE =MB 199             Parametrierfehlerbyte MB199
0002C      :
0003E      :BE
```

Nach einem Netzein benötigt die CPU24x NET ca. 15s für den Hochlauf. Wenn während dieser Anlaufphase vom AG synchronisiert wird, erhält man im Parametrierfehlerbyte PAFE einen Fehler zurückgemeldet. Diese Meldung verschwindet, sobald die CP-Baugruppe hochgelaufen ist.

Der Timer in diesem Baustein wird anfangs auf 20s gesetzt. Wenn innerhalb dieser Zeit nicht ordentlich synchronisiert wird, wird die Bearbeitung gestoppt.

Die einstellbare Blockgröße entnehmen Sie der nachfolgenden Tabelle.

| Blockgröße | CP-Blockgröße in Byte |
|---|---|
| 0 | Voreingestellte Blockgröße verwenden |
| 1 | 16 |
| 2 | 32 |
| 3 | 64 |
| 4 | 128 |
| 5 | 256 |
| 6 | 512 |
| 255 | 512 |

### Zyklus

Im Zyklusprogramm OB1 sind die Sende und Empfangsbausteine SEND und RECEIVE für den Sende- und Empfangsanstoß zu parametrieren. Die eigentliche Datenübertragung wird von den Bausteinen SEND ALL und RECEIVE ALL durchgeführt.

Bei einer rein passiven Verbindung sind nur die Bausteine SEND ALL bzw. RECEIVE ALL erforderlich.

Zur Sicherung der Datenübertragung sollten Sie Kontrollinstanzen durch Auswertung des Anzeigenwortes integrieren.

# Beispiel zur Projektierung

## Übersicht und Voraus-setzungen

Dieses Kapitel soll in den Umgang mit den Bussystemen H1 und TCP/IP für das System 200V einführen. Dabei wird insbesondere der Einsatz der Parametriersoftware WinNCS von VIPA im Vordergrund stehen.

Ziel dieses Kapitels ist es, ein kleines Kommunikationssystem zwischen einer CPU 24x NET und einer CP 143 H1 / TCP/IP aufzubauen, das auf einfache Weise die Kontrolle der Kommunikationsvorgänge erlaubt.

Kenntnisse über die CP-Hantierungsbausteine sind erforderlich. CP-Hantierungsbausteine sind Standardfunktionsbausteine. Sie ermöglichen die Nutzung der Kommunikationsfunktionen durch Programme in den Automatisierungsgeräten.

Für die Durchführung der Beispiele sollten Sie mindestens die folgenden technischen Einrichtungen besitzen:

1 CPU 24x NET von VIPA,
1 Rack-135U von VIPA,
1 CPU 928 B/C von VIPA,
1 CP 143 H1 / TCP/IP-Baugruppe von VIPA.

Übertragungsstrecke bestehend aus:

- 2 Buskabel ,
- 1 HUB.

Software-Pakete

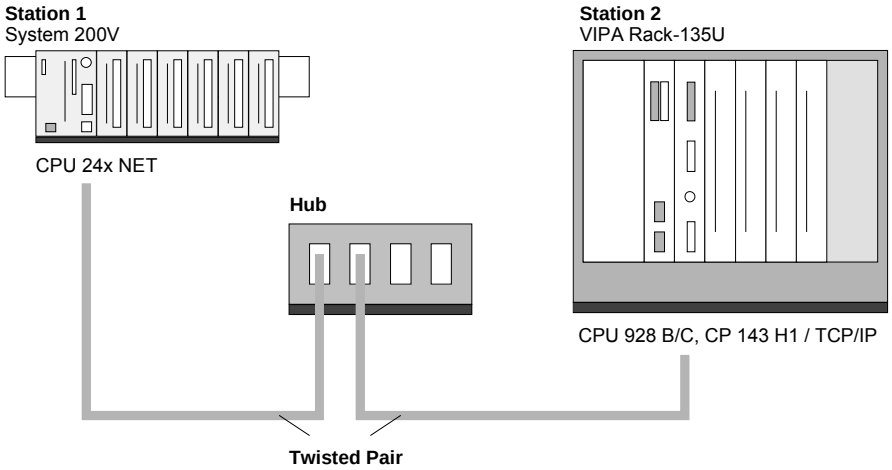
- Parametriersoftware WinNCS von VIPA
- MC5 Programmierpaket für SPS-Programmierung von VIPA

In folgendem Beispiel wird die zugrunde gelegte Aufgabenstellung näher erläutert. Zur Realisierung des Beispiels ist die Programmierung der zwei Automatisierungsgeräte in der Sprache STEP®5 von Siemens sowie die Parametrierung der Kommunikationsprozessoren unter WinNCS erforderlich.

## Aufgaben-stellung

Dem einführenden Beispiel für die Anwendung des H1- bzw. TCP/IP-Protokolls wird eine Kommunikationsaufgabe zugrunde gelegt, die im folgenden näher erläutert wird:

## Anlagenstruktur für das H1-Protokoll




## Aufgaben für die AGs

### Aufgaben für Station 1 (CPU 24x NET)

- Im Datenbaustein DB 11 werden die Datenworte DW 0 bis DW 99 alle 100ms übertragen.
- Das Datenwort DW 0 im DB 11 dient dabei als Telegrammzähler. Es wird nur dann inkrementiert, wenn der vorhergegangene Sendeauftrag korrekt (fertig ohne Fehler) abgearbeitet wurde. Über die restlichen Datenworte (DW 1 bis DW 99) könnten Nutzdaten übertragen werden.
- SEND wird mit der Auftragsnummer A-Nr. = 11 und dem Kacheloffset SSNR = 0 parametriert.
- Die Quellparameter sind direkt zu parametrieren.

### Aufgaben für AG2 (CPU 928 B/C)

- Die vom AG1 gesendeten Daten sollen im AG2 empfangen und im Datenbaustein DB 12 hinterlegt werden. Hierzu wird der Hantierungs-baustein RECEIVE benutzt.
- Der Hantierungsbaustein RECEIVE wird mit der Auftragsnummer A-NR = 12 und dem Kacheloffset SSNR = 0 parametriert.
- Die Zielparameter sind im Datenbaustein DB 12 ab Datenwort DW 0 hinterlegt.
- Zu dem angegebenen Kacheloffset SSNR = 0 muss eine passende Parametrierung der Baugruppe CP143 H1 / TCP/IP vorgenommen werden. Sie ist in diesem Beispiel identisch mit der Parametrierung des CP-Teils von Station 1.

Die Aufgabenstellung und die erforderlichen Voreinstellungen sind somit umrissen. Weitere Einzelheiten zur Parametrierung der Hantierungs-bausteine gehen aus den Programmen hervor. Die dazu passende Para-metrierung der CPs unter H1 bzw. TCP/IP ist ausführlich beschrieben.

## Projektierung unter WinNCS

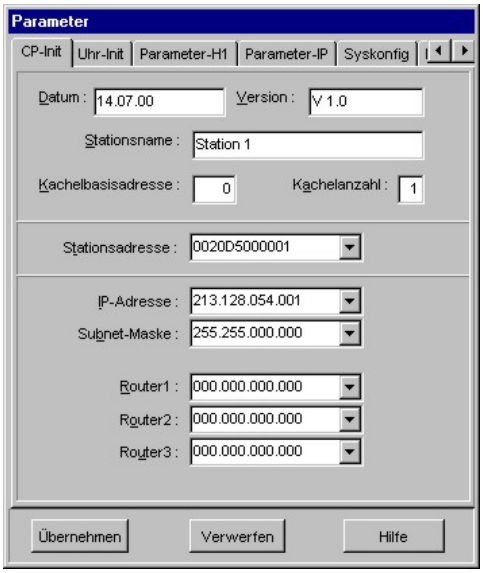
Die Projektierung der beiden CPs findet ausschließlich unter WinNCS statt. Starten Sie WinNCS und legen Sie ein Projekt mit der Funktionsgruppe "Ethernet_H1" an. Die Vorgehensweise ist bei beiden Stationen die gleiche. Sie unterscheidet sich nur in den einzustellenden Parametern und gliedert sich in folgende 3 Teile:

- CP-Grundprojektierung,
- Verbindungsbausteine projektieren,
- Projektierdaten in den CP übertragen.

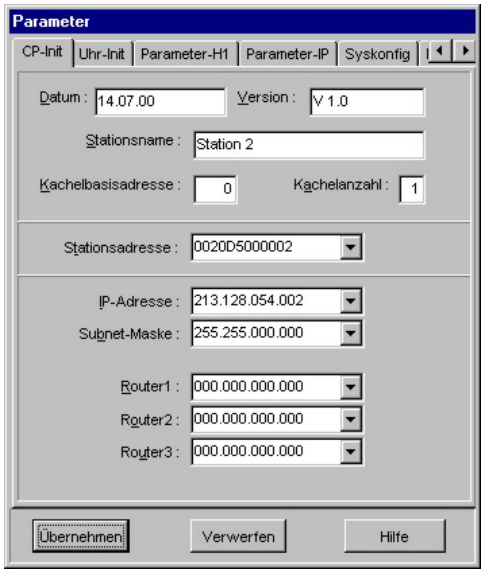
## CP-Grund-projektierung

Fügen Sie zwei Stationen ein und stellen Sie folgende Werte ein:

Station 1




Station 2




Die erforderlichen Stationsadressen erhalten Sie von Ihrem Systemadministrator.

Führen Sie gegebenenfalls noch weitere Einstellungen in den Parametrierfenstern durch. Näheres hierzu erfahren Sie von Ihrem Systemadministrator.

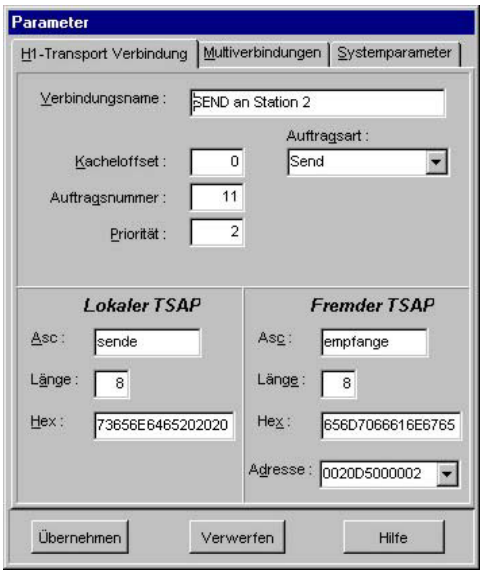
**Verbindungsbausteine projektieren**

*H1-Verbindungen projektieren*

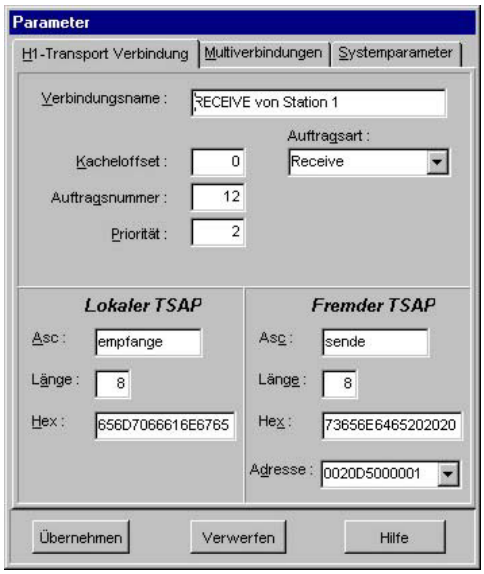
Zur Projektierung Ihrer H1-Verbindungen fügen Sie unterhalb der Stationen mit eine H1-Transportverbindung ein und geben für die Stationen folgende Parameter ein:

H1-Verbindungen

Station 1




Station 2



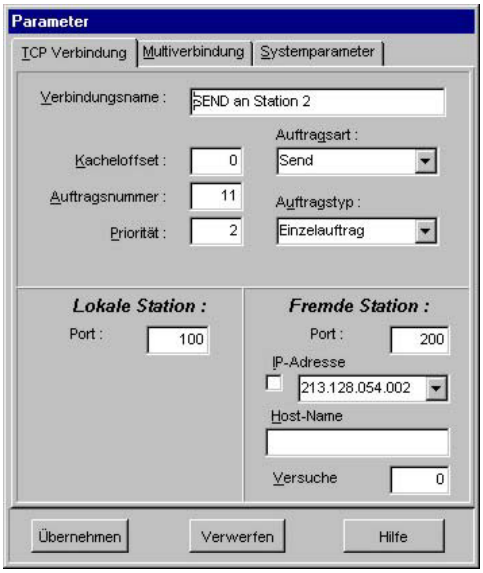

**alternativ - oder**

*TCP/IP-Verbindungen projektieren*

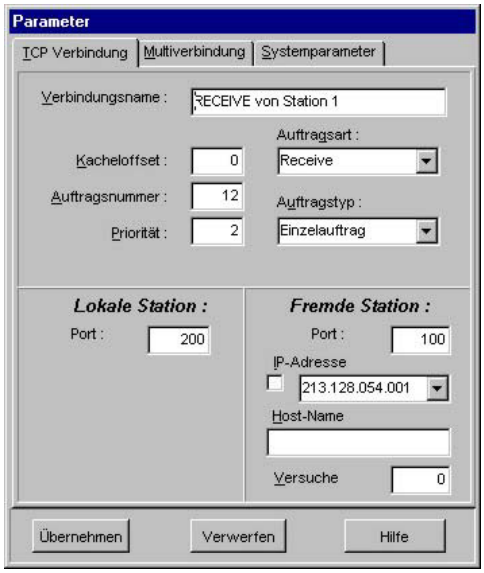
Zur Projektierung Ihrer TCP/IP-Verbindungen fügen Sie unterhalb der Stationen mit eine TCP-Verbindung ein und geben folgende Parameter ein:

TCP/IP-Verbindungen

Station 1




Station 2



**Projektierdaten in den CP übertragen**

Ihr Netzwerkfenster sollte folgenden Inhalt zeigen:

Sie können online über das Netzwerk Ihre Projektierung in die entsprechenden CPs übertragen. Bauen Sie hierzu wie weiter oben gezeigt die Anlagenstruktur auf und fahren Sie beide CPs hoch.

Bringen Sie für die Datenübertragung beide CPs in den STOP-Zustand.

Den CP-Teil der CPU 24x NET können Sie über die "Modultransfer-Funktionen" in den STOP-Zustand bringen.

Den CP 143 H1 / TCP/IP bringen Sie mit dem RUN/STOP-Schalter in STOP.

Markieren Sie im Netzwerkfenster von WinNCS die Station, deren Projektierdaten Sie übertragen wollen und rufen Sie mit die Modultransfer-Funktion auf.

Stellen Sie unter INIT die IP-Adresse Ihres Ziel-CPs ein und übertragen Sie mit PC => CP die zugehörigen Projektierdaten in den CP.

Nach der Übertragung ist bei der CPU 24x NET der Befehl CP=>FLASH zwingend erforderlich, ansonsten wird Ihr Programm wieder gelöscht.

Wiederholen Sie die oben gezeigten Vorgänge für die Station 2. Hier ist eine Übertragung in das Flash-ROM nicht erforderlich.

Nähere Angaben Datenübertragung finden Sie im Handbuch WinNCS unter "Modultransfer".

Die Projektierung auf CP-Seite ist jetzt abgeschlossen. Auf der Folgeseite finden Sie die Programmierung der SPS-Seite.

## SPS-Programme für die CPUs

Die SPS-Programmierung ist in diesem Beispiel protokollunabhängig und kann für H1 und TCP/IP verwendet werden.

### Programm für Station 1 (CPU 24x NET)

***Synchronisation der Schnittstellen***

Im Anlauf eines AGs muss jede benutzte Schnittstelle eines CPs mittels des Hantierungsbausteins SYNCHRON synchronisiert werden. Da dies für jede Anlaufart des AGs gilt, ist je nach Notwendigkeit im:

- OB 21 für Neustart,
- OB 22 für Wiederanlauf nach Netzspannungsausfall,

die erforderliche Anzahl von SYNCHRON-Bausteinen abzusetzen.

Im Funktionsbaustein NETZEIN (FB 22) wird überprüft, ob die Synchronisation fehlerfrei durchgeführt wurde. Trat ein Fehler auf, wird das Programm durch einen Stoppbefehl (STP) abgebrochen.

Folgende Bausteine sind in Station 1 zu übertragen.

Operationsbausteine OB 21, OB 22:

```
Netzwerk 1
;                                    Anlauf nach Netzein
00000      :
00002      :SPA FB 22                SYNCHRON aufrufen
      NAME #NETZEIN
00006      :
00008      :BE
```

Funktionsbaustein FB 22:

```
      BSTNAME #NETZEIN
      BIB     #2085
0000E      :
00022 SYN  :SPA FB 249          Synchron-Baustein
      NAME #SYNCHRON
      SSNR =KY 0,0              SSNR bzw. Kachelbasis 0
      BLGR =KY 0,6              Blockgrösse 6 (512 Byte)
      PAFE =MB 199              Parametrierfehlerbyte MB199

0002E      :UN  M 199.0         wenn kein Fehler, dann Ende
00030      :BEB
00038      :L   KH 2222         Fehlerkennung in Akku laden
0003C      :STP                 CPU in Stop schalten
0003E      :BE
```

Nach NETZEIN bzw. WIEDERANLAUF benötigt der CP ca. 18s für den Hochlauf. Erfolgt innerhalb des Synchronbausteins nach maximal 30s keine Synchronisierung, wird eine Fehlermeldung zurückgegeben.

***Zyklus-OB, -FB und DB***

Der Sendeanstoß in Station 1 erfolgt mittels eines SEND-Hantierungsbausteins. Dieser wird im Funktionsbaustein FB 1 aufgerufen.

Der Sprung in den FB 1 ist der erste Befehl im Zyklus-Organisationsbaustein OB 1. Der Sendeauftrag wird wie folgt parametriert.

**Zyklus-Operationsbaustein OB 1**:

```
Netzwerk 1
;                     Zyklus
00001      :SPA FB 1            FB1 Sendeteil
      NAME #FB1
      TIME =KT 10.0            Sendetimer  10 entspr. 100ms
00002      :O  M 0.0
00004      :ON M 0.0            Verknüpfungsergebnis 1
00006      :SPA FB 244          Send Baustein
      NAME #SEND
      SSNR =KY 0,0             SSNR bzw. Kachelbasis 0
      A-NR =KY 0,0             Auftrag-Nr. 0 für All-Auftrag
      ANZW =MW 190             Anzeigewort im MW190
      QTYP =KC                 irrelevant
      DBNR =KY 0,0             irrelevant
      QANF =KF +0              irrelevant
      QLAE =KF +0              irrelevant
      PAFE =MB 199             Parametrierfehlerbyte MB199

00012      :SPA FB 244          Receive Baustein
      NAME #RECEIVE
      SSNR =KY 0,0             SSNR bzw. Kachelbasis 0
      A-NR =KY 0,0             Auftrag-Nr. 0 für All-Auftrag
      ANZW =MW 194             Anzeigewort im MW194
      ZTYP =KC                 irrelevant
      DBNR =KY 0,0             irrelevant
      ZANF =KF +0              irrelevant
      ZLAE =KF +0              irrelevant
      PAFE =MB 198             Parametrierfehlerbyte MB198

00020      :
00030      :BE
```

**Datenbaustein DB 11**:

```
BAUSTEIN#DB11
      BIB     #9095
00000:       KH    = 0000                    1. Datum
Telegrammzähler
00001:   100 (                               weitere 100
Datenworte
00001:       KH    = 1111
00002:       )
```

Für die Anstoßhäufigkeit eines SEND-Auftrags ist der parametrierte Zeitwert beim FB1-Aufruf maßgebend. Der Timer ist in diesem Beispiel auf 100ms programmiert. Damit stößt das Beispielprogramm den SEND-Auftrag im 100ms-Takt an.

Vor jedem SEND-Aufruf, der tatsächlich ein Telegramm absendet, wird das Datenwort DW 0 des Datenbausteins DB 11 inkrementiert. Dies geschieht im Funktionsbaustein FB 1.

Zusammen mit DW 0 können noch 99 Nutzdaten übertragen werden.

**Funktionsbaustein FB 1**:

```
      BSTNAME #FB1
      BIB     #19075
      BEZ     #TIME  D:KT
00010       :
00014       :SPA FB 247          Control Baustein
      NAME #CONTROL
      SSNR =KY 0,0               Kachelbasis 0
      A-NR =KY 0,11              Aufrags-Nr. 11
      ANZW =MW 0                 Anzeigewort ins MW0
      PAFE =MB 189               Parametrierfehlerbyte MB189
00020       :
00022       :O   M 1.1           solange Auftrag läuft
00024       :O   T 11            oder Timer läuft
00026       :BEB                 keine weitere Bearbeitung
00028       :
0002A       :LW  =TIME           parametrierten Zeitwert laden
0002C       :U   M 0.0           und Timer neu starten
0002E       :UN  M 0.0
00030       :SV  T 11            VKE 0 Timer Start
00032       :O   M 0.0
00034       :ON  M 0.0
00036       :SV  T 11            VKE 1 Timer Start
00038       :
0003A       :U   M 1.3           wenn fertig mit Fehler,
0003C       :SPB =SEND           dann Zähler nicht erhöhen
0003E       :
00040       :A   DB 11           Sende-DB
00042       :L   DW 0            Telegrammzähler
00044       :L   KB 1            im DW0, das auch gesendet wird
00046       :+F
00048       :T   DW 0
0004A       :
0004C SEND  :
0004E       :SPA FB 244          Send Baustein
      NAME #SEND
      SSNR =KY 0,0               Schnittstellen-Nr. 0
      A-NR =KY 0,11              Auftrag-Nr. 11
      ANZW =MW 0                 Anzeigewort im MW0
      QTYP =KC DB                Daten senden aus DB
      DBNR =KY 0,11              DB-Nummer 11
      QANF =KF +0                ab Datenwort 0
      QLAE =KF +100              100 Elemente (DWs)
      PAFE =MB 99
00062       :
00064       :BE
```

FB 1 ist ein einfacher Sendebaustein mit Auswertung der Anzeigenwort-flags:

- Auftrag läuft,
- Fertig mit Fehler.

**Programm für Station 2 (CPU 928 B/C)**

***Synchronisation der Schnittstellen***

Wie bei Station 1 ist auch hier mittels des Synchron-Bausteins jede benutzte Schnittstelle in folgenden OBs zu synchronisieren:

- OB 20 für Neustart,
- OB 21 für manuellen Wiederanlauf,
- OB 22 für Wiederanlauf nach Netzspannungsausfall.

**Operationsbausteine: OB 20, OB 21, OB 22**:

```
Netzwerk 1
;                                  Anlauf nach Netzein
00000      :
00002      :SPA FB 22              Wartezeit und synchron
      NAME #NETZEIN
00006      :
00008      :BE
```

Im Funktionsbaustein NETZEIN (FB 22) wird überprüft, ob die Synchronisation fehlerfrei durchgeführt wurde. Trat ein Fehler auf, so wird das Programm durch einen Stoppbefehl (STP) abgebrochen.

Nach einem Netzein benötigt der CP143 H1 / TCP/IP eine Hochlaufzeit von ca. 18s. Wenn während dieser Anlaufphase vom AG synchronisiert wird, erhält man im Parametierfehlerbyte einen Fehler zurückgemeldet. Diese Meldung verschwindet, sobald die CP-Baugruppe hochgelaufen ist.

Der Timer in diesem Baustein wird anfangs auf 20s gesetzt. Wenn innerhalb dieser Zeit nicht ordentlich synchronisiert wird, wird die Bearbeitung gestoppt.

**Funktionsbaustein FB 22**:

```
      BSTNAME #NETZEIN
      BIB     #2085
0000E      :
00010      :L   KT 20.2           max. Wartezeit  20s
00014      :U   M 0.0
00016      :UN  M 0.0
00018      :SV  T 43              VKE 0 für neg. Flanke
0001A      :O   M 0.0
0001C      :ON  M 0.0
0001E      :SV  T 43              VKE 1 zum Start des Timers
00020      :
00022 SYN  :SPA FB 125            Synchron-Baustein
      NAME #SYNCHRON
      SSNR =KY 0,0                SSNR bzw. Kachelbasis 0
      BLGR =KY 0,6                Blockgrösse 6 (512 Byte)
      PAFE =MB 199                Parametrierfehlerbyte MB199
0002C      :
0002E      :UN  M 199.0           wenn kein Fehler, dann Ende
00030      :BEB
00032      :
00034      :U   T 43              solange Timer läuft
00036      :SPB =SYN              Synchronisierung versuchen
00038      :L   KH 2222           Fehlerkennung in Akku laden
0003C      :STP                   AG in Stop schalten
0003E      :BE
```

### Zyklus-OB, -FB und -DB

Die vom Station 1 gesendeten Daten werden mittels des Hantierungsbausteins RECEIVE in Station 2 empfangen. Der entsprechende Aufruf wird im Organisationsbaustein OB 1 abgesetzt.

**Zyklus-Operationsbaustein OB 1**:

```
Netzwerk 1
;                      Zyklus
00001      :SPA FB 2            FB2 Empfangsteil
      NAME #FB2
00002      :O  M 0.0
00004      :ON M 0.0            Verknüpfungsergebnis 1
00006      :SPA FB 126          Send All
      NAME #SEND-A
      SSNR =KY 0,0             SSNR bzw. Kachelbasis  0
      A-NR =KY 0,0             A-Nr. 0 für  All-Auftrag
      ANZW =MW 190             Anzeigewort MW190
      PAFE =MB 199             Parametrierfehlerbyte MB199
00012      :
00014      :SPA FB 127          Receive All
      NAME #REC-A
      SSNR =KY 0,0             SSNR bzw. Kachelbasis 0
      A-NR =KY 0,0             A/Nr. 0 für All/Auftrag
      ANZW =MW 194             Anzeigewort MW194
      PAFE =MB 198             Parametrierfehlerbyte MB198
00020      :
00030      :BE
```

**Datenbaustein DB 12**:

```
BAUSTEIN#DB12
      BIB     #9095
;                           Empfangs-Datenbaustein (A-Nr. 12)
00000:       KH    = 0000   1. Datum    Sende-Telegrammzähler
00001:   100 (              weitere 100 Datenworte
00001:       KH    = 0000
00002:       )
```

**Funktionsbaustein FB 2:**

```
      BSTNAME #FB2
      BIB     #19075
0000C       :
0000E       :SPA FB 123          Control Baustein
      NAME #CONTROL
      SSNR =KY 0,0               Schnittstellen-Nr. 0
      A-NR =KY 0,12              Auftrags-Nr. 12
      ANZW =MW 4                 Anzeigewort MW4
      PAFE =MB 199
0001A       :
0001C       :UN  M 5.0           wenn keine Daten da,
0001E       :BEB                 dann Ende
00020       :
00028       :L   MW 12           Empfangszähler erhöhen
0002A       :L   KB 1
0002C       :+F
0002E       :T   MW 12
00030       :
00032 REC   :
00034       :SPA FB 121          Receive Baustein
      NAME #RECEIVE
      SSNR =KY 0,0               Schnittstellen-Nr. 0
      A-NR =KY 0,12              Auftrags-Nr. 12
      ANZW =MW 4                 Anzeigenwort MW4
      ZTYP =KC DB                Datenablage in Datenbaustein
      DBNR =KY 0,12              DB-Nummer 12
      ZANF =KF +0                ab DW0
      ZLAE =KF +100              Länge 100 Elemente
      PAFE =MB 199
00048       :
0004A       :BE
```

FB 2 ist ein einfacher Empfangsbaustein mit Auswertung der Anzeigenwortflags:

- Handshake sinnvoll,
- Fertig mit Fehler.

## Beobachtung der Übertragung mit dem MC5-Paket

Zur Beobachtung von Verbindungsaufträgen sind sowohl die Baugruppen als auch die CPUs zu programmieren.

Als Ausgangspunkt werden parametrierte Baugruppen und urgelöschte Automatisierungsgeräte, deren RUN/STOP-Schalter in der Grundstellung STOP steht, vorausgesetzt.

Laden Sie nun die vorher beschriebenen SPS-Programme in die beiden CPUs und starten Sie diese, indem Sie jeweils den RUN/STOP-Schalter in die Stellung RUN bringen. Die Kommunikation zwischen den beiden Baugruppen ist aufgebaut. Angezeigt wird dies durch die COMM-LED.

Starten Sie das MC5-Paket von VIPA auf Ihrem PC und führen Sie zur Beobachtung des Sendeauftrags die folgenden Schritte aus:

- Rufen Sie vom Hauptmenü aus das Testmenü auf (F8).
- Springen Sie mit "Steu.Var" (F5) in das Untermenü "Steuern Variablen." Die Bildschirmmaske enthält in der ersten Kopfzeile die Information über die angezeigte Funktion, den Belegungsgrad der aktuellen Seite (in %) und die aktuelle Seitenzahl (von 1 bis 20). Im weiteren ist die Maske dreispaltig aufgebaut. In der Spalte "Operand" werden die Prozess-variablen durch Editieren oder Laden einer Variablenliste eingetragen. In der Spalte "Format" können, in Abhängigkeit vom Operanden-Typ, Änderungen vorgenommen werden.
- Tragen Sie unter "Operand" die entsprechenden Datenbausteine (hier DB 11) und Datenworte (hier DW 0 bis DW 99) ein.
- Ab DW 1 können Nutzdaten eingetragen werden. Fahren Sie hierzu mit dem Cursor auf "Wert" und tragen Sie einen zu übertragenden Wert ein. Im Beispiel erhält "Wert" den Eintrag "1111".

- Drücken Sie F1 (Start). Die eingegebenen Werte werden übertragen und folgendes Ausgabefenster geöffnet:

```
MC5 198600 Byte frei  TEST - Steuern von Variablen
 Steuern Variablen       Belegt:  35%        Seite: 1      AG läuft

 Operand                   Format    Wert                                  Ueb
 KOM                       Senden  ANZW:MW0  Sendebereich DB11 ab DW0
 KOM                       Sendezaehler im DW0
 MW  0                     KM        00000000 00000100
 DB  11
 DW  0                     KH        A61E
 DW  1                     KH        1111
 DW  2                     KH        1111
 DW  3                     KH        1111
 DW  99                    KH        1111

   F1    ¦   F2    ¦   F3    ¦   F4    ¦   F5    ¦   F6    ¦   F7    ¦   F8
         ¦         ¦         ¦         ¦Editieren¦         ¦         ¦
```

Für die Anzeige des Receive-Auftrags verbinden Sie den PC mit dem AG2. Führen Sie die gleichen Schritte durch, wie bei AG1 und tragen Sie folgende Operanden ein:

```
MC5 148968 Byte frei  TEST - Steuern von Variablen
 Steuern Variablen       Belegt:  35%        Seite: 2      AG läuft

 Operand                   Format    Wert                                  Ueb
 KOM                       Empfang ANZW: MW4 Empfangsbereich DB12 ab DW0
 KOM                       DW0: Zaehler, der auf Senderseite inkrementiert wird
 MW  4                     KM        00000100 00010100
 DB  12
 DW  0                     KH        861A
 DW  1                     KH        1111
 DW  2                     KH        1111
 DW  98                    KH        1111
 DW  99                    KH        1111

   F1    ¦   F2    ¦   F3    ¦   F4    ¦   F5    ¦   F6    ¦   F7    ¦   F8
         ¦         ¦         ¦         ¦Editieren¦         ¦         ¦
```

Nach Betätigen von "Start" wird die Datenmaske aktualisiert, d.h. die übertragenen Daten werden angezeigt.

Stehen zwei PCs zur Verfügung, kann die Beobachtung parallel in beiden AGs erfolgen. Bei korrekter Programmierung sind die Datenworte DW 0 bis DW 99 in den Datenbausteinen DB 11 bzw. DB 12 gleich und werden praktisch gleichzeitig (lediglich um die Zeit der Übertragung zueinander verschoben) geändert.

# Anlaufverhalten

Nach dem Einschalten der Stromversorgung durchlaufen CPU und CP ihre BIOS-Routinen (Hardware-, Treiberinitialisierung und Speichertest).

Während die CPU die Module am Rückwandbus ermittelt und das Anwenderprogramm lädt, beginnt der CP-Teil mit der Kachelverwaltung.

Nach ca. 15s wartet der CP auf die Synchronisation mit der CPU. In diesem Zustand ist der Datenverkehr mit dem AG gesperrt und wird erst mit der Synchronisation freigegeben.

Die Hochlaufzeit der CPU 24x NET einschließlich CP-Teil beträgt ca. 18s.

**Status nach CP-Anlauf**

Die CPU24x NET vollzieht bei jedem Zustandswechsel von STOP nach RUN, sowie von RUN über STOP nach RUN einen Kalt-/Warmstart. Alle bisher aufgebauten Verbindungen werden gelöscht und nach dem Boot des CP-Job erneut aufgebaut.

Solche Zustandswechsel-Anforderungen können drei Quellen haben:

- Neusynchronisation eines bereits synchronen CPs durch die Synchron-HTBs der CPU (Warmstart),
- STOP/START-Funktion des Parametrierwerkzeugs WinNCS (Warmstart),
- Reset-All (Warmstart).

# Systemeigenschaften der CPU 24x NET

**Hinweis**

Systemeigenschaften eines CPs sind nicht als Einschränkungen zu sehen bzw. einem Fehlverhalten gleichzustellen, vielmehr sind gewisse Funktionalitäten nicht zu erreichen, oder aus Sicht des Gesamtsystems gewollt.

**Allgemein**

Die Hochlaufzeit des CP-Teils der CPU 24x NET beträgt ca. 18 Sekunden. Diese Hochlaufzeit wird im integrierten SYNCHRON-Baustein berücksichtigt (Wartezeit 30s).

**Nur bei H1 beachten**

- Aufträge der Priorität 0/1 können maximal so viele Daten senden und/oder empfangen, wie dies durch den SYNCHRON-HTB definiert wurde. Die Aufträge dieser Priorität werden nicht geblockt. Daraus ergibt sich eine maximale Daten-Transferrate von 512 Byte pro Auftrag bei einer Blockgröße von 255 (siehe hierzu auch Blockgröße).
- RECEIVE-Aufträge, die auf die Kommunikationsart Broadcast abgebildet werden, können bei einem schnellen zyklischen Sender nicht alle Datentelegramme empfangen. Die nicht empfangenen Telegramme werden verworfen.

**Nur bei TCP/IP beachten**

- Die Jokerlänge (-1, 0xFFFF) ist bei der ORG-Format-Längenangabe nicht zulässig, d.h. der Anwender muss eine genaue Länge seiner zu empfangenen Daten definieren.
- Aufträge der Priorität 1 können maximal so viele Daten senden und/oder empfangen, wie dies durch den SYNCHRON-HTB definiert wurde. Die Aufträge dieser Priorität werden nicht geblockt. Daraus ergibt sich eine maximale Daten-Transferrate von 512 Byte pro Auftrag bei einer Blockgröße von 255 (siehe hierzu auch Blockgröße).
- RECEIVE-Aufträge, die auf die Kommunikationsart UDP abgebildet werden, können bei einem schnellen zyklischen Sender nicht alle Datentelegramme empfangen. Die nicht empfangenen Telegramme werden verworfen.

Der Protokollstack TCP/IP besitzt einen globalen Pufferpool, in dem sich die Empfangs- und Sende-Puffer befinden. Hier kann es zu Systemkollisionen kommen wenn:

- Daten für einen Empfangsauftrag nicht abgeholt werden. Nach geraumer Zeit werden die Ressourcen knapp und die anderen Verbindungen führen nach einer definierten Zeit Verbindungsabbrüche durch.
  Eine ausgewogene Kommunikation kann allerdings erst wieder aufgenommen werden, wenn die Empfangspuffer der einen Verbindung freigegeben (Verbindungsabbruch) bzw. mit den RECEIVE-HTBs die Daten abgeholt wurden.

- Ein oder mehrere zyklische Sender einen CP belasten. Es werden bei Ressourcen-Engpässen ebenfalls Verbindungsabbrüche vom CP initiiert.
- Ein Sender zwei oder mehrere Telegramme sendet und der Empfänger hatte noch nicht die Möglichkeit es abzuholen, so würde es bei einem dem Empfänger unbekannten Datentyp im Empfänger zu Datenkollisionen kommen. Dies wird aber auf der CP-Seite verhindert. Die SPS-Applikation fordert eine definierte Empfangsgröße, wobei die Jokerlänge nicht zulässig ist. Die Empfangsgrößen der Prio1 - RECEIVE-Aufträge ist implizit durch die eingestellte Blockgröße definiert (16, 32, 64, 128, 256, 512 Byte).
- Zur Gewährleistung einer hundertprozentigen Datenübertragungssicherheit, empfiehlt VIPA bei der Datenübertragung Quittungstelegramme auf Anwenderebene einzusetzen.

# Kopplung mit Fremdsystemen

**ORG-Format**

Das Organisationsformat ist die Kurzbeschreibung einer Datenquelle bzw. eines Datenziels in SPS-Umgebung. Die verwendbaren ORG-Formate sind in der nachfolgenden Tabelle aufgelistet.

Der ORG-Block ist bei READ und WRITE optional.

Die ERW-Kennung ist bei der Adressierung von Datenbausteinen relevant. In diesem Fall wird hier die Datenbaustein-Nummer eingetragen. Die Anfangsadresse und Anzahl adressieren den Speicherbereich und sind im HIGH-/LOW- Format abgelegt (Motorola - Adressformat)

| Beschreibung | Typ | Bereich |
|---|---|---|
| ORG-Kennung | BYTE | 1..x |
| ERW-Kennung | BYTE | 1..255 |
| Anfangsadresse | HILOWORD | 0..y |
| Anzahl | HILOWORD | 1..z |

In der nachfolgenden Tabelle sind die verwendbaren ORG-Formate aufgelistet. Die "Länge" darf nicht mit -1 (FFFFh) angegeben werden.

*ORG-Kennung 01h-04h*

| AG-Bereich | DB | MB | EB | AB |
|---|---|---|---|---|
| ORG-Kennung | 01h | 02h | 03h | 04h |
| Beschreibung | Quell-/Zieldaten aus/in Datenbaustein im Hauptspeicher. | Quell-/Zieldaten aus/in Merkerbereich. | Quell-/Zieldaten aus/in Prozessabbild der Eingänge (PAE). | Quell-/Zieldaten aus/in Prozessabbild der Ausgänge (PAA). |
| DBNR<br><br>erlaubter Bereich: | DB, aus dem die Quelldaten entnommen werden bzw. in den die Zieldaten transferiert werden.<br><br>1...255 | irrelevant | irrelevant | irrelevant |
| Anfangsadresse Bedeutung<br><br>erlaubter Bereich: | DW-Nr., ab der die Daten entnommen bzw. eingeschrieben werden.<br><br>1...2047 | MB-Nr., ab der die Daten entnommen bzw. eingeschrieben werden.<br><br>0...255 | EB-Nr., ab der die Daten entnommen bzw. eingeschrieben werden.<br><br>0...127 | AB-Nr., ab der die Daten entnommen bzw. eingeschrieben werden.<br><br>0...127 |
| Anzahl Bedeutung<br><br>erlaubter Bereich: | Länge des Quell-/Zieldatenblocks in Worten.<br><br>1...2048 | Länge des Quell-/Zieldatenblocks in Bytes.<br><br>1...256 | Länge des Quell-/Zieldatenblocks in Bytes.<br><br>1...128 | Länge des Quell-/Zieldatenblocks in Bytes.<br><br>1...128 |

*ORG-Kennung 05h-08h*

| AG-Bereich | PB | ZB | TB | BS |
|---|---|---|---|---|
| ORG-Kennung | 05h | 06h | 07h | 08h |
| Beschreibung | Quell-/Zieldaten aus/in Peripheriebaugruppen. Bei Quelldaten Eingabebaugruppen, bei Zieldaten Ausgabebaugruppen. | Quell-/Zieldaten aus/in Zählerzellen. | Quell-/Zieldaten aus/in Zeitenzellen. | Quell-/Zieldaten aus/in Systemdatenbereich (BS und BS'). |
| DBNR | irrelevant | irrelevant | irrelevant | irrelevant |
| Anfangsadresse Bedeutung | PB-Nr., ab der die Daten entnommen bzw. eingeschrieben werden. | ZB-Nr., ab der die Daten entnommen bzw. eingeschrieben werden. | TB-Nr., ab der die Daten entnommen bzw. eingeschrieben werden. | Nummer des BS-Worts ab dem die Daten entnommen bzw. eingeschrieben werden. |
| erlaubter Bereich: | 0...127 digitale Periph.<br>128...255 anal. Periph. | 0...255 | 0...255 | 0...511 |
| Anzahl Bedeutung | Länge des Quell-/Zieldatenblocks in Bytes. | Länge des Quell-/Zieldatenblocks in Worten (Zählerzelle = 1 Wort). | Länge des Quell-/Zieldatenblocks in Worten (Zählerzelle = 1 Wort). | Länge des Quell-/Zieldatenblocks in Worten. |
| erlaubter Bereich: | 1...256 | 1...256 | 1...256 | 1...128 |

*ORG-Kennung 09h-11h*

| AG-Bereich | AS | DX*) | QB*) | SM |
|---|---|---|---|---|
| ORG-Kennung | 09h | 0Ah | 0Bh | 10h |
| Beschreibung | Quell-/Zieldaten aus/in absolut adressierten Speicherzellen. | Quell-/Zieldaten aus/in erweiterten Datenbaustein. | Quell-/Zieldaten aus/in Periph.-baugr. im erw. Peripheriebereich. Bei Quelldaten Eingabebaugruppe, bei Zieldaten Ausgabebaugruppe. | Quell-/Zieldaten aus/in Sonder Merkerbereich. |
| DBNR | irrelevant | DX, aus dem die Quelldaten entnommen werden bzw. in den die Zieldaten abgelegt werden. | irrelevant | irrelevant |
| erlaubter Bereich: | | 1...255 | | |
| Anfangsadresse Bedeutung | absolute Anfangsadresse, ab der die Daten entnommen bzw. eingeschrieben werden. | DW-Nr. ab der die Daten entnommen bzw. eingeschrieben werden. | PB-Nr. ab der die Daten entnommen bzw. eingeschrieben werden. | Sondermerker-Nr. ab der die Daten entnommen bzw. eingeschrieben werden. |
| erlaubter Bereich: | 0h...FFFFh | 0...2047 | 0...511 | 0...1023 |
| Anzahl Bedeutung | Länge des Quell-/Zieldatenblocks in Worten. | Länge des Quell-/Zieldatenblocks in Worten. | Länge des Quell-/Zieldatenblocks in Bytes. | Länge des Quell-/Zieldatenblocks in Bytes. |
| erlaubter Bereich: | 1...32767 | 1...2048 | 1...256 | 1...1024 |

*) Die Bereiche DX und QB sind in der CPU 24x NET nicht vorhanden.

**Aufbau SPS-Header**

Bei READ und WRITE generiert der CP SPS-Header für Anforderungs-und Quittungstelegramme. Diese Header sind in der Regel 16Byte lang und haben folgende Struktur:

**bei WRITE**

Anforderungstelegramm

| | |
|---|---|
| Systemkennung | ="S" |
| | ="5" |
| Länge.Header | =16d |
| Kenn.OP-Code | =01 |
| Länge OP-Code | =03 |
| **OP-Code** | **=03** |
| ORG-Block | =03 |
| Länge ORG-Block | =08 |
| ORG-Kennung | |
| DBNR | |
| Anfangs- | H |
| adresse | L |
| Länge | H |
| | L |
| Leerblock | =FFh |
| Länge Leerbl. | =02 |
| Daten bis zu 64K jedoch nur wenn Fehler-Nr.=0 | |

Quittungstelegramm

| | |
|---|---|
| Systemkennung | ="S" |
| | ="5" |
| Länge.Header | =16d |
| Kenn.OP-Code | =01 |
| Länge OP-Code | =03 |
| **OP-Code** | **=04** |
| Quittungsblock | =0Fh |
| Länge Q-Block | =03 |
| Fehler Nr. | =Nr. |
| Leerblock | =FFh |
| Länge Leerblock | =07 |
| | |
| frei | |
| | |
| | |
| | |

**bei READ**

Anforderungstelegramm

| | |
|---|---|
| Systemkennung | ="S" |
| | ="5" |
| Länge.Header | =16d |
| Kenn.OP-Code | =01 |
| Länge OP-Code | =03 |
| **OP-Code** | **=05** |
| ORG-Block | =03 |
| Länge ORG-Block | =08 |
| ORG-Kennung | |
| DBNR | |
| Anfangs- | H |
| adresse L | |
| Länge | H |
| | L |
| Leerblock | =FFh |
| Länge Leerbl. | =02 |

Quittungstelegramm

| | |
|---|---|
| Systemkennung | ="S" |
| | ="5" |
| Länge.Header | =16d |
| Kenn.OP-Code | =01 |
| Länge OP-Code | =03 |
| **OP-Code** | **=06** |
| Quittungsblock | =0Fh |
| Länge Q-Block | =03 |
| Fehler Nr. | =Nr. |
| Leerblock | =FFh |
| Länge Leerblock | =07 |
| | |
| frei | |
| | |
| | |
| | |
| Daten bis zu 64K jedoch nur wenn Fehler-Nr.=0 | |

**SEND / RECEIVE vom Typ TRADA**

TRADA steht für **Tr**ansparenten **D**aten**a**ustausch. Beim transparenten Datenaustausch können Nutzdaten mit unterschiedlicher Länge übertragen werden. Den zu übertragenden Nutzdaten wird ein 16-Byte-Header vorangestellt, der die Länge der Nutzdaten angibt.

Bei Einsatz von TRADA ist im SPS-Anwenderprogramm die Angabe einer Jokerlänge bei der Längenangabe zulässig.

Durch Eingabe von -1 als Längenangabe im RECEIVE-FB (Parameter: ZLAE) stellen Sie eine variable Nutzdatenlänge ein (Jokerlänge). Bei Einsatz einer Jokerlänge wird die jeweilige Datenlänge aus dem TRADA-Header übernommen.

Der nachfolgend beschriebene Header wird bei TRADA-Funktionalität einem SEND-Auftrag vorangestellt und von RECEIVE ausgewertet.

SEND vom Typ TRADA
OP-Code = 07

| | |
|---|---|
| Systemkennung | ="S" |
| | ="5" |
| Länge.Header | =16d |
| Kenn.OP-Code | =01 |
| Länge OP-Code | =03 |
| **OP-Code** | **=07** |
| ORG-Block | =03 |
| Länge ORG-Block | =08 |
| ORG-Kennung | |
| DBNR (**irrelevant**) | |
| Anfangs- | H |
| adresse | L |
| **Länge** | **H** |
| | **L** |
| Leerblock | =FFh |
| Länge Leerbl. | =02 |
| Daten bis zu 64K jedoch nur wenn Fehler-Nr.=0 | |

→ Längenangabe der Nutzdaten

**Längenangabe**

In der Längenangabe steht die Anzahl der Bytes eines Datenblocks.

Wird mit einer Blockgröße von 6 synchronisiert (512Byte), erfolgt die Längenangabe in Worten.

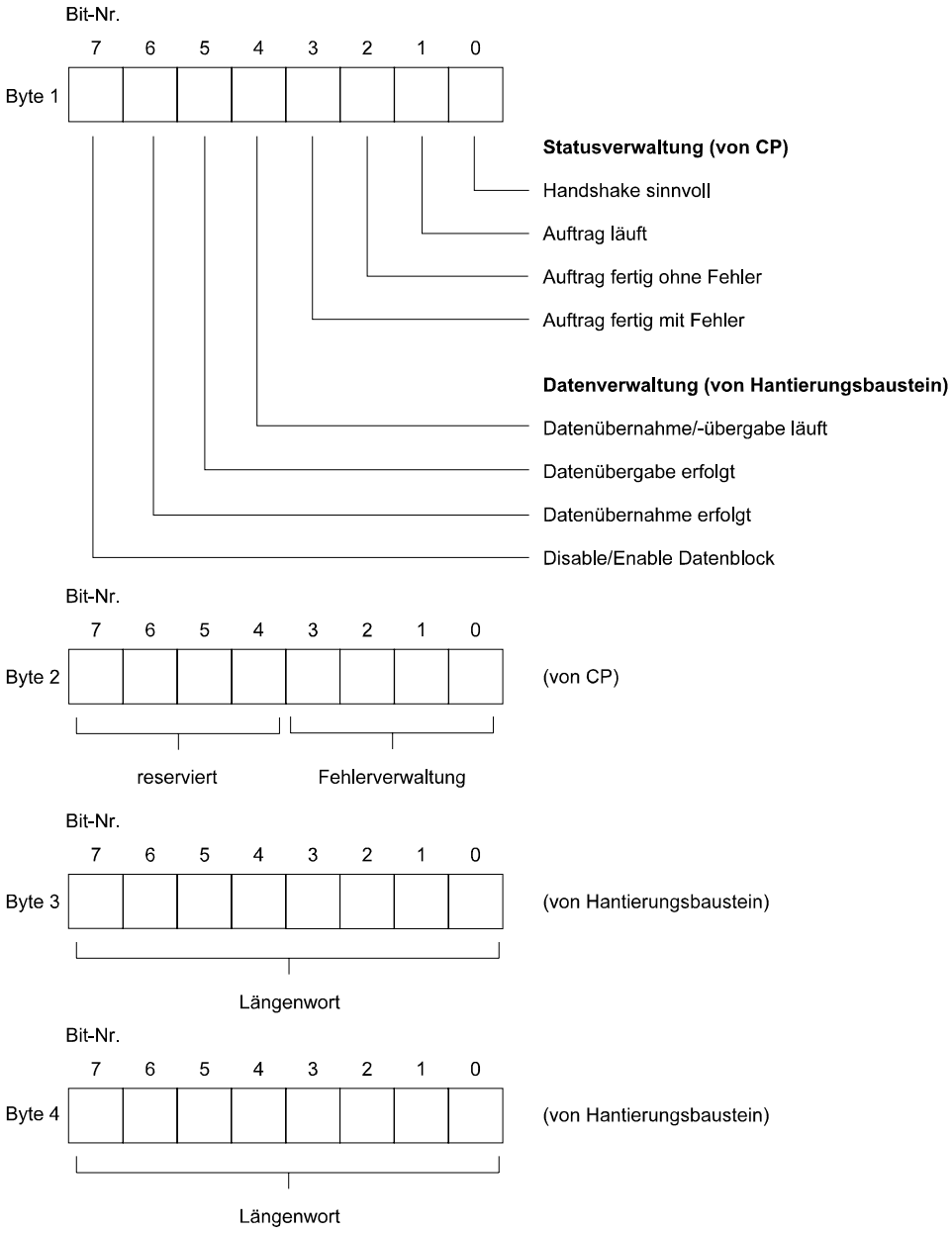
# Status- und Fehleranzeigen

Status und Fehleranzeigen liefern die Hantierungsbausteine:

- über das Anzeigenwort ANZW (Informationen zur Auftragsbearbeitung),
- über das Parametrierfehlerbyte PAFE (Anzeige einer fehlerhaften Auftragsparametrierung).

**Inhalt und Aufbau Anzeigenwort ANZW**

Das Anzeigenwort hat den folgenden prinzipiellen Aufbau:



**Statusverwaltung Byte 1 Bit 0 ... Bit 3**

Hier ist verschlüsselt, ob ein Auftrag bereits gestartet ist, ob Fehler aufgetreten sind oder ob der Auftrag, z.B. weil die virtuelle Verbindung nicht besteht, gesperrt ist.

**Bit 0: Handshake sinnvoll**

Setzen: Durch die Anschaltung entsprechend der "Löschen"-Anzeige im Auftragsstatusbit. Handshake sinnvoll (=1) wird beim RECEIVE-Baustein genutzt. (Telegramm vorhanden bei PRIO 1 oder RECEIVE-Anstoß möglich bei PRIO 2/3).

Auswerten: Durch den RECEIVE-Baustein: Nur wenn das Bit gesetzt, leitet der RECEIVE den Handshake mit dem CP ein. Durch die Anwendung: Für RECEIVE-Anfrage (Abfrage, ob Telegramm vorhanden bei PRIO 1).

**Bit 1: Auftrag läuft**

Setzen: Durch die Anschaltung, wenn Auftrag an CP erteilt.

Löschen: Durch die Anschaltung, wenn ein Auftrag abgearbeitet ist (z.B. Quittung eingetroffen).

Auswerten: Durch die Hantierungsbausteine: Ein neuer Auftrag wird nur erteilt, wenn der "alte" Auftrag abgearbeitet ist.

Durch den Anwender: um zu erfahren, ob das Triggern eines neuen Auftrags sinnvoll ist.

**Bit 2: Auftrag fertig ohne Fehler**

Setzen: Durch die Anschaltung, wenn der entsprechende Auftrag ohne Fehler abgeschlossen wurde.

Löschen: Durch die Anschaltung, wenn der Auftrag erneut ausgelöst wird.

Auswerten: Durch den Anwender zur Prüfung, ob der Auftrag fehlerlos abgeschlossen wurde.

**Bit 3: Auftrag fertig mit Fehler**

Setzen: Durch die Anschaltung, wenn der entsprechende Auftrag mit Fehler abgeschlossen wurde. Die Fehlerursache ist dann im High-Teil des Anzeigenwortes verschlüsselt.

Löschen: Durch die Anschaltung, wenn der Auftrag erneut ausgelöst wird.

Auswerten: Durch den Anwender: Zur Prüfung, ob der Auftrag mit Fehler abgeschlossen, wurde. Ist die Kennung "Auftrag fertig mit Fehler" gesetzt, steht im High-Byte des Anzeigenwortes die Fehlerursache.

**Datenverwaltung Byte 1 Bit 4 ... Bit 7**

Hier ist verschlüsselt, ob der Datentransfer für den Auftrag noch läuft oder ob die Datenübergabe bzw. Datenübernahme bereits abgeschlossen ist. Mit dem Bit "Enable / Disable" kann der Datentransfer für den Auftrag gesperrt werden. (Disable = 1; Enable = 0).

**Bit 4: Datenübernahme / Datenübergabe läuft**

Setzen: Durch die Hantierungsbausteine SEND, RECEIVE, wenn die Übergabe/Übernahme für einen Auftrag begonnen wurde, z.B. wenn Daten über die ALL-Funktion (DMA-Ersatz) ausgetauscht werden, der Anstoß jedoch mit SEND-DIREKT erfolgte.

Löschen: Durch die Hantierungsbausteine SEND, RECEIVE, wenn der Datenaustausch für einen Auftrag beendet ist (letzter Teilblock übertragen).

Auswerten: Durch den Anwender: Während der Datenübertragung CP<< >>AG darf der Anwender den Datensatz eines Auftrags nicht mehr verändern. Bei PRIO 0/1 Aufträgen ist dies unkritisch, da hierbei der Datenaustausch in einem Baustein-Durchlauf erledigt werden kann. Größere Datenmengen können jedoch nur in Blöcken übertragen werden, wobei diese Blockung über mehrere AG-Zyklen verteilt wird. Zur Wahrung der Datenkonsistenz ist zu prüfen ob der Datenblock gerade übertragen wird, bevor dessen Inhalt geändert wird.

**BIT 5: Datenübergabe erfolgt**

Setzen: Durch den Hantierungsbaustein SEND, wenn die Datenübergabe für einen Auftrag erfolgt ist.

Löschen: Durch den Hantierungsbaustein SEND, wenn für einen neuen Auftrag (neue Triggerung) mit dem Transfer der Daten begonnen wurde.

Durch den Anwender: Wenn die Auswertung erfolgte (Flankenbildung).

Auswerten: Durch den Anwender: Mit diesem Bit ist zu ermitteln, ob der Datensatz für einen Auftrag schon auf den CP übertragen wurde bzw. wann ein neuer Datensatz für einen laufenden Auftrag (z.B. zyklische Übertragung) bereitgestellt werden kann.

**Bit 6: Datenübernahme erfolgt**

Setzen: Durch den Hantierungsbaustein RECEIVE, wenn die Übernahme von Daten für einen Auftrag abgeschlossen wurde.

Löschen: Durch den Hantierungsbaustein RECEIVE, wenn für einen neuen Auftrag (neue Triggerung) mit dem Transfer der Daten ins AG begonnen wurde. Durch den Anwender, wenn die Auswertung erfolgt (Flankenbildung).

Auswerten: Durch den Anwender: Mit diesem Bit kann der Anwender ermitteln, ob der Datensatz eines Auftrags schon auf das AG übertragen wurde bzw. wann ein neuer Datensatz für einen laufenden Auftrag ins AG transferiert wurde.

**Bit 7: Disable / Enable Datenblock**

| | |
|---|---|
| Setzen: | Durch den Anwender, um das Beschreiben eines Bereichs durch den RECEIVE-Baustein bzw. das Auslesen aus einem Bereich durch den SEND-Baustein zu verhindern (nur beim 1. Datenblock). |
| Löschen: | Durch den Anwender, um den zugehörigen Datenbereich freizugeben. |
| Auswerten: | Durch die Hantierungsbausteine SEND und RECEIVE. Ist das Bit 7 gesetzt, führen die Bausteine keinen Datenverkehr durch, sondern melden dem CP den Fehler. |

**Fehlerverwaltung Byte 2 Bit 0 ... Bit 3**

Hier werden die Fehleranzeigen des Auftrags angezeigt. Diese Fehleranzeigen sind nur gültig, wenn auch gleichzeitig das Bit "Auftrag fertig mit Fehler" im Statusbit gesetzt ist.

Folgende Fehlermeldungen können ausgegeben werden:

0 **kein Fehler**

Sollte das Bit "Auftrag fertig mit Fehler" gesetzt sein, so hat der CP143 H1 / TCP/IP die Verbindung neu aufbauen müssen wie z.B. nach einem Neustart oder RESET.

1 **falscher Q/ZTYP am HTB**

Auftrag wurde mit falscher TYP-Kennung parametriert.

2 **Bereich im AG nicht vorhanden**

Beim Anstoß des Auftrags wurde ein falscher DB (DBNR) parametriert.

3 **Bereich im AG zu klein**

Die Summe aus Q/ZANF und Q/ZLAE überschreitet die Bereichsgrenzen. Die Bereichsgrenze wird bei Datenbausteinen durch die Bausteingröße bestimmt. Bei Merkern, Zeiten, Zählern usw. ist die Bereichsgröße AG-abhängig.

4 **QVZ-Fehler im AG**

Mit dem Quell- bzw. Zielparameter wurde ein Bereich im AG angegeben, dessen Speicher defekt oder nicht bestückt ist. Der QVZ-Fehler kann nur bei Q/ZTYP AS, PB, QB oder bei Speicherdefekten auftreten.

5 **Fehler beim Anzeigenwort**

Das parametrierte Anzeigenwort kann nicht bearbeitet werden. Dieser Fehler tritt auf, wenn mit ANZW ein Datenwort bzw. Doppelwort angegeben wurde, das sich nicht oder nicht mehr in dem spezifizierten Datenbaustein befindet d.h. DB zu klein oder nicht vorhanden.

6 **kein gültiges ORG-Format**

Das Datenziel bzw. die Datenquelle ist weder beim Hantierungsbaustein (Q/TYP="NN") noch im Verbindungsbaustein angegeben.

7 **Reserviert**

8 **keine freien Transportverbindungen**

Die Transportverbindungskapazitäten sind überschritten. Löschen Sie unnötige Verbindungen.

9 **Remote-Fehler**

Bei einem READ/WRITE-Auftrag ist ein Fehler im Kommunikationspartner aufgetreten.

A **Verbindungsfehler**

Die Verbindung für einen Auftrag ist nicht bzw. noch nicht aufgebaut. Der Fehler verschwindet, sobald eine Verbindung aufgebaut werden kann. Sind alle Verbindungen des CPs unterbrochen, so deutet dies auf eine Defekt der Baugruppe oder des Buskabels hin. Der Fehler kann auch durch eine fehlerhafte Parametrierung ausgelöst werden, wie z.B. fehlerhafte Adressierung.

B **Handshakefehler**

Dies kann ein Systemfehler sein oder die Datenblockgröße ist zu groß gewählt.

C **Anstoßfehler**

Zum Anstoß des Auftrags wurde ein falscher Hantierungsbaustein benutzt, oder ein zu großer Datenblock übergeben.

D **Abbruch nach RESET**

Hier handelt es sich um eine Betriebsmeldung. Bei Priorität 1 und 2 ist die Verbindung unterbrochen und wird neu aufgebaut, sobald sich der Kommunikationspartner auf eine neue Verbindung eingestellt hat. Bei Priorität 3 Verbindungen ist die Verbindung gelöscht, ein neuer Anstoß ist möglich.

E **Auftrag mit Urladefunktion**

Dies ist eine Betriebsmeldung. Der Auftrag ist ein READ/WRITE-PASSIV und kann vom AG aus nicht gestartet werden.

F **Auftrag nicht vorhanden**

Der angesprochene Auftrag ist nicht auf dem CP143 H1 / TCP/IP parametriert. Dieser Fehler kann auftreten wenn SSNR/A-NR Kombination im Hantierungsbaustein falsch oder kein Verbindungsbaustein eingetragen ist.

Die Bits 4 bis 7 von Byte 2 sind für Erweiterungen reserviert.

**Längenwort Byte 3 ... Byte 4**

Im Längenwort hinterlegen die Hantierungsbausteine (SEND, RECEIVE) die Menge für den entsprechenden Auftrag bereits transferierten Daten, d.h. bei Empfangsaufträgen die bereits empfangene Datenmenge, bei Sendeaufträgen die bereits gesendete Datenmenge.

Beschreiben: Durch SEND, RECEIVE während des Datenaustausches. Das "Längen-Wort" wird errechnet aus:

**akt. Übertragungsanz. + Anz. bereits getauschter Daten**

Löschen: Durch Überschreiben bzw. mit jedem neuen SEND, RECEIVE, FETCH. Wenn Bit "Auftrag fertig ohne Fehler" bzw. "Datenübergabe/-übernahme erfolgt" gesetzt ist, steht im "Längen-Wort" die aktuelle Quell- bzw. Ziellänge.

Wenn das Bit "Auftrag fertig mit Fehler" gesetzt ist, beinhaltet das Längenwort die bis zum Fehlerfall übertragene Datenanzahl.

**Wichtige Status- und Fehleranzeigen der CPU 24x NET**

Im folgenden sind wichtige Status- und Fehlermeldungen aufgeführt, die im "Anzeigenwort" erscheinen können. Die Darstellung hierbei erfolgt in "HEX"-Mustern, wie man sie auch mit der Status / Steuern-Var-Testfunktion mit dem PG im AG beobachten kann. Das Zeichen X steht für "nicht bestimmt" bzw. für "irrelevant"; Nr. ist die Fehlernummer.

**Mögliche Anzeigenworte**

*Anzeigenwort: X F X A*

Die Fehlerkennung "F" besagt, dass der entsprechende Auftrag auf dem CP 143 nicht definiert ist. Die Statuskennung A bewirkt, dass der Auftrag gesperrt ist ( für SEND / FETCH und RECEIVE).

*Anzeigenwort: X A X A*

Die Fehlerkennung "A" zeigt an, dass die Verbindung des Kommunikationsauftrags nicht bzw. noch nicht aufgebaut ist. Mit der Statuskennung "A" ist sowohl der SEND als auch der RECEIVE und FETCH gesperrt.

*Anzeigenwort: X 0 X 8*

Die Verbindung ist neu aufgebaut (z.B. nach einem CP-Neuanlauf), der SEND ist freigegeben (SEND-Kommunikationsauftrag).

*Anzeigenwort: X 0 X 9*

Die Verbindung ist neu aufgebaut, der RECEIVE ist freigegeben. (RECEIVE-Kommunikationsauftrag).

*Anzeigenwort: X 0 2 4*

Der SEND ist ohne Fehler abgearbeitet worden, die Daten wurden übertragen.

*Anzeigenwort: X 0 4 5*

Der RECEIVE ist ohne Fehler abgearbeitet worden, die Daten sind auf dem AG angekommen.

*Anzeigenwort: X 0 X 2*

Der SEND-, RECEIVE-, READ- bzw. WRITE-Auftrag läuft. Bei SEND hat sich der Partner noch nicht auf den RECEIVE eingestellt. Bei RECEIVE hat der Partner noch kein SEND abgesetzt.

**Wichtige Anzeigenwort-Zustände**

Anzeigen bei SEND

| Zustand unter H1 | Prio 0/1 | Prio 2 | Prio 3/4 |
|---|---|---|---|
| Zustand unter TCP/IP | Prio 1 | Prio 2 | Prio 3 |
| nach Neustart | 0 A 0 A | 0 A 0 A | 0 0 0 8 |
| nach Verbindungsaufbau | X 0 X 8 | X 0 X 8 | ..... |
| nach Anstoß | X 0 X 2 | X 0 X 2 | X 0 X 2 |
| fertig ohne Fehler | X 0 2 4 | X 0 2 4 | X 0 2 4 |
| fertig mit Fehler | X Nr X 8 | X Nr X 8 | X Nr X 8 |
| nach RESET | X D X A | X D X A | X D X 8 |

Anzeigen bei RECEIVE

| Zustand unter H1 | Prio 0/1 | Prio 2 | Prio 3/4 |
|---|---|---|---|
| Zustand unter TCP/IP | Prio 1 | Prio 2 | Prio 3 |
| nach Neustart | 0 A 0 A | 0 A 0 A | 0 0 0 1 |
| nach Verbindungsaufbau | X 0 X 4 | X 0 0 9 | ..... |
| nach Anstoß | X 0 X 2 | X 0 X 2 | X 0 X 2 |
| Telegramm da | X 0 X 1 | ..... | ..... |
| fertig ohne Fehler | X 0 4 1 | X 0 4 5 | X 0 4 5 |
| fertig mit Fehler | X Nr X8 | X Nr X 9 | X Nr X 9 |
| nach RESET | X D X A | X D X A | X D X 9 |

Anzeigen bei READ/WRITE-AKTIV

| Zustand unter H1 | Prio 0/1 | Prio 2 | Prio 3/4 |
|---|---|---|---|
| Zustand unter TCP/IP | Prio 1 | Prio 2 | Prio 3 |
| nach Neustart | | 0 A 0 A | |
| nach Verbindungsaufbau | | X 0 0 8 | |
| nach Anstoß | | X 0 X 2 | |
| READ fertig | | X 0 4 4 | |
| WRITE fertig | | X 0 2 4 | |
| fertig mit Fehler | | X Nr X 8 | |
| nach RESET | | X D X A | |

Anzeigen bei SEND bzw. RECEIVE bei Bausteinkennung "NN"
bei SEND (keine Quell- / Zielparameter)

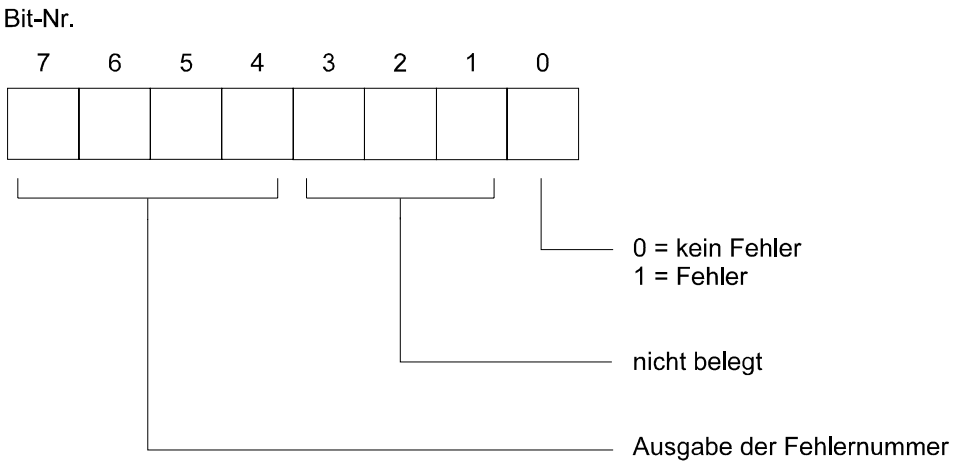
| Zustand unter H1 | Prio 0/1 | Prio 2 | Prio 3/4 |
|---|---|---|---|
| Zustand unter TCP/IP | Prio 1 | Prio 2 | Prio 3 |
| fertig ohne Fehler | X 0 0 4 | X 0 0 4 | X 0 0 4 |

bei RECEIVE

| Zustand unter H1 | Prio 0/1 | Prio 2 | Prio 3/4 |
|---|---|---|---|
| Zustand unter TCP/IP | Prio 1 | Prio 2 | Prio 3 |
| fertig ohne Fehler | X 0 0 4 | X 0 0 5 | X 0 0 5 |

**PAFE Parametrier-fehlerbyte**

Das PAFE wird gesetzt, wenn der Hantierungsbaustein einen Parametrier-fehler erkannt hat.




**Fehlernummern:**

| | |
|---|---|
| 0 | kein Fehler |
| 1 | falsches ORG-Format |
| 2 | Bereich nicht vorhanden |
| 3 | Bereich zu klein |
| 4 | QVZ-Fehler |
| 5 | falsches Anzeigenwort |
| 6 | keine Quell-/ Zielparameter bei SEND / RECEIVE ALL |
| 7 | Schnittstelle nicht vorhanden |
| 8 | Schnittstelle unklar |
| 9 | Schnittstelle überlastet |
| A | frei |
| B | unzulässige Auftragsnummer ANR |
| C | Schnittstelle nicht quittiert bzw. freigegeben |
| D | nicht belegt |
| E | nicht belegt |
| F | nicht belegt |

# Testprogramm für TCP/IP-Verbindungen

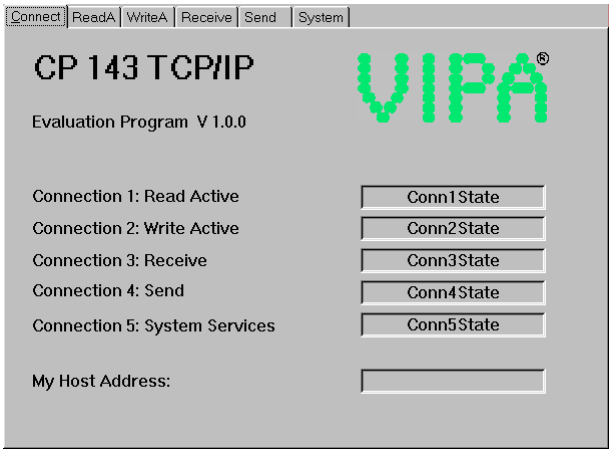
Im Lieferumfang von WinNCS befindet sich Testprogramm TCPTest. Mit diesem Testprogramm können Sie einfache TCP/IP-Verbindungen aufbauen und analysieren.

Das Testprogramm lautet TCPTEST.EXE und befindet sich im Lieferumfang von WinNCS. Nach der Installation von WinNCS finden Sie TCPTest im Verzeichnis NCS.

Der Einsatz dieses Testprogramms soll hier kurz gezeigt werden.

Starten Sie hierzu TCPTEST.EXE. Das Testprogramm wird aufgerufen und meldet sich mit folgendem Fenster:

**Startbildschirm**




**Registerblätter**

Das Menü ist in Form von Registerblättern aufgebaut. Durch Anklicken mit der linken Maustaste kommt das ausgewählte Dialogfenster in den Vordergrund.

**Registerblätter**

| | |
|---|---|
| *Connect* | Fenster mit Statusanzeige der Verbindungen und der eigenen IP-Adresse. |
| *ReadA* | Parametrierfenster für READ AKTIV-Verbindung (FETCH). |
| *WriteA* | Parametrierfenster für WRITE AKTIV-Verbindung. |
| *Receive* | Parametrierfenster für RECEIVE-Auftrag. |
| *Send* | Parametrierfenster für SEND-Auftrag. |
| *System* | Steuerfenster für Statusabfrage und RUN/STOP-Schaltung des CPs. |

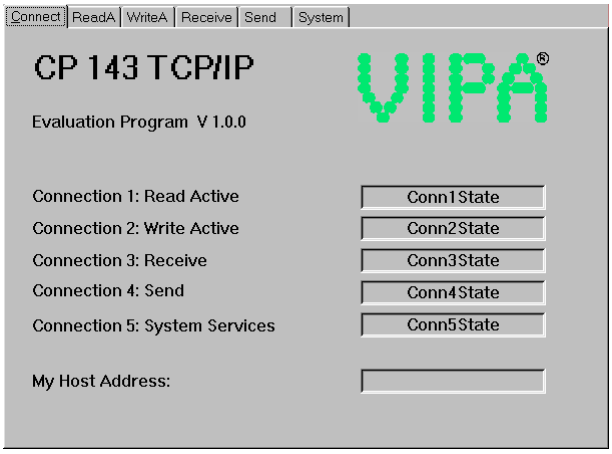
**Kontextmenü (rechte Maustaste)**

In jedem Registerblatt haben Sie die Möglichkeit ein Kontextmenü zu aktivieren. Die Aktivierung erfolgt mit der rechten Maustaste.

Durch Betätigung der rechten Maustaste haben Sie immer Zugriff auf ein Kontextmenü, das folgende Auswahl bietet:

| | |
|---|---|
| *Save All* | Alle Parameter speichern. |
| *Save Conn 1* bis *Save Conn5* | Speichert die jeweilige Verbindung. |
| *Save Win Pos* | Speichert die aktuelle Fensterposition. |
| *Show Hints* | Wenn Sie mit der Maus auf ein Eingabefeld oder auf eine Schaltfläche fahren so wird, wenn "Show Hints" angewählt ist, ein Kurzhilfetext eingeblendet. |

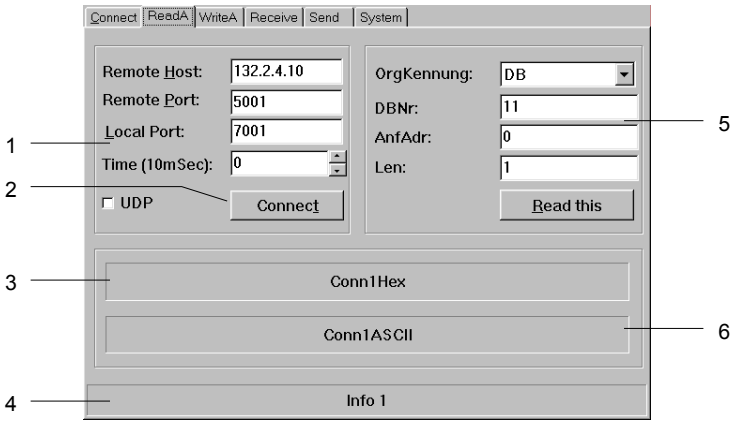
**Connect-Register (Status)**




Das Fenster zeigt den Status aller in diesem Programm parametrierbaren Verbindungen an. Sie können hier auf einen Blick erkennen, welche Verbindungen stabil und welche instabil sind. Sobald sich in einem Register ein Status ändert, wird dies in diesem Fenster angezeigt.

Hier finden Sie auch zur Kontrolle Ihre eigene IP-Adresse.

### ReadA-Register




[1] Verbindungsdaten
[2] Verbindung aufbauen
[3] hexadezimale Darstellung
[4] Informationsfenster für Verbindungsstatus
[5] Quelldaten
[6] ASCII-Darstellung der empfangenen Daten

Das Fenster bietet Ihnen die Möglichkeit zur Parametrierung einer aktiven Leseverbindung.

Neben den Daten für den Verbindungsaufbau geben Sie hier die Quelle an, aus der Daten gelesen werden sollen.

#### Eingabefelder

| | |
|---|---|
| *Remote Host* | IP-Adresse der Station, aus der Daten gelesen werden sollen. |
| *Remote Port* | Verbindungsadresse der fremden Station. |
| *Local Port* | Verbindungsadresse der eigenen Station. Zur Vereinfachung können Sie für Remote und Local die gleiche Adresse verwenden. |
| *Time (10mSec)* | Einstellbarer Zeitintervall für zyklisches Lesen. |
| *OrgKennung* | Typ des Quellbausteins. |
| *DBNr* | Nummer des Quellbausteins. |
| *AnfAdr* | Anfangsadresse des Quellbausteins. |
| *Len* | Wortlänge des Quellbausteins. |

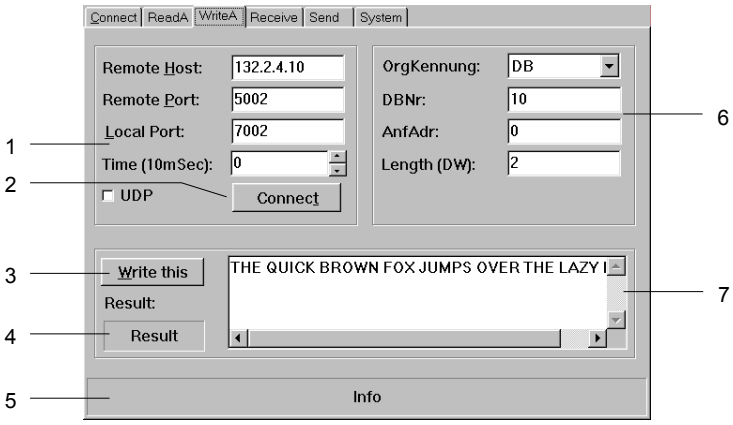
#### Kontrollkästchen

| | |
|---|---|
| *UDP* | Hier wählen Sie eine ungesicherte Kommunikation an. Bei der ungesicherten Kommunikation wird mit keiner virtuellen Verbindung gearbeitet. Auf diese Weise können Sie nur UDP-Telegramme anzeigen. |

#### Schaltflächen

| | |
|---|---|
| *Connect* | Die Verbindung wird aufgebaut und für den Lesevorgang vorbereitet. |
| *Read this* | Über diese Verbindung werden die angeforderten Daten gelesen. |

### WriteA-Register




[1] Verbindungsdaten
[2] Verbindung aufbauen
[3] Daten über Verbindung schicken
[4] Ergebnis-Code des Schreibauftrags
[5] Informationsfenster für Verbindungsstatus
[6] Quelldaten
[7] ASCII-Text, der an den CP zu übertragen ist

Hier aktivieren Sie eine aktive Schreibverbindung.

Analog zum READ AKTIV-Befehl geben Sie hier neben den Daten für den Verbindungsaufbau den Zielbaustein an, in den die Daten zu übertragen sind.

#### Eingabefelder

| | |
|---|---|
| *Remote Host* | IP-Adresse der Station, in die Daten zu schreiben sind. |
| *Remote Port* | Verbindungsadresse der fremden Station. |
| *Local Port* | Verbindungsadresse der eigenen Station. Zur Vereinfachung können Sie für Remote und Local die gleiche Adresse verwenden. |
| *Time (10mSec)* | Einstellbarer Zeitintervall für zyklisches Schreiben. Für das zyklische Schreiben muss der Timerwert mindestens 5 betragen. |
| *OrgKennung* | Typ des Zielbausteins. |
| *DBNr* | Nummer des Zielbausteins. |
| *AnfAdr* | Anfangsadresse des Zielbausteins. |
| *Len* | Wortlänge des Zielbausteins. |

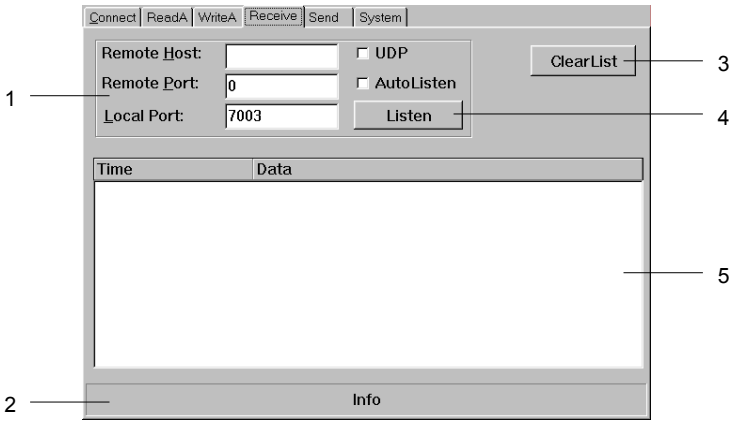
#### Kontrollkästchen

| | |
|---|---|
| *UDP* | Hier wählen Sie eine ungesicherte Kommunikation an. Bei der ungesicherten Kommunikation wird mit keiner virtuellen Verbindung gearbeitet. Das Telegramm wird als UDP-Telegramm geschickt. |

#### Schaltflächen

| | |
|---|---|
| *Connect* | Die Verbindung wird aufgebaut und für den Schreibvorgang vorbereitet. |
| *Write this* | Über die durch *Connect* aufgebaute Verbindung werden die im ASCII-Feld eingegebenen Daten in den CP geschrieben. |

### Receive-Register




[1] Verbindungsdaten
[2] Informationsfenster für Verbindungsstatus
[3] Empfangsliste löschen
[4] Telegramme auflisten
[5] Liste der empfangenen Telegramme

In diesem Dialogfenster können Sie den Empfang von Telegrammen eines bestimmten Hostrechners parametrieren.

#### Eingabefelder

*Remote Host* IP-Adresse der Station, in die Daten zu schreiben sind.

*Remote Port* Verbindungsadresse der fremden Station.

*Local Port* Verbindungsadresse der eigenen Station. Zur Vereinfachung können Sie für Remote und Local die gleiche Adresse verwenden.

#### Kontrollkästchen

*UDP* Hier wählen Sie eine ungesicherte Kommunikation an. Bei der ungesicherten Kommunikation wird mit keiner virtuellen Verbindung gearbeitet. Es werden nur UDP-Telegramme empfangen und angezeigt.

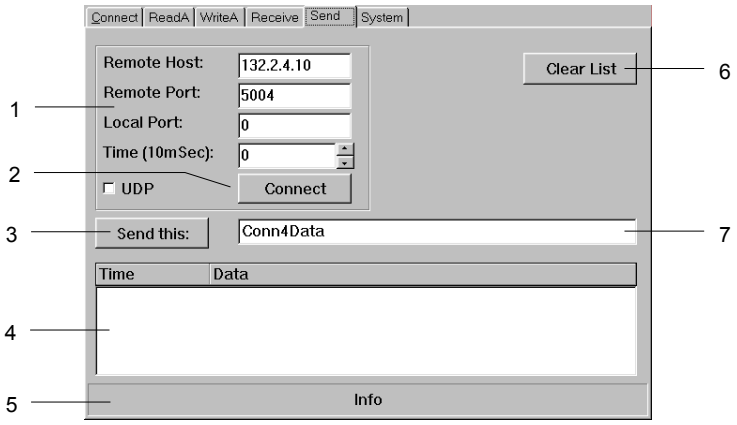
*AutoListen* Durch Anwahl von "AutoListen" springt das Programm in den Empfangsmodus. Es wird jedes empfangene Telegramm des Remote-CPs in der Liste angezeigt. Verbindungsunterbrechungen werden erkannt und angezeigt, das Programm bleibt aber empfangsbereit. Sobald die Verbindung wieder steht, wird die Telegrammauflistung fortgesetzt.

#### Schaltflächen

*Listen* Die empfangenen Telegramme werden aufgelistet. Sobald Sie auf die Schaltfläche "STOP" klicken oder die Verbindung unterbrochen wird, wird die Auflistung beendet. Sie können auch durch die Eingabe neuer Verbindungsparameter die Auflistung beenden.

*ClearList* Die Empfangsliste wird gelöscht und die Auflistung in der obersten Zeile der Liste fortgesetzt.

### Send-Register




[1] Verbindungsdaten
[2] Verbindung aufbauen
[3] Daten über Verbindung schicken
[4] Liste der gesendeten Telegramme
[5] Informationsfenster für Verbindungsstatus
[6] Telegrammsliste löschen
[7] ASCII-Text, der an den CP zu übertragen ist

In diesem Dialogfenster können Sie ein Telegramm an einen spezifizierten Hostrechner schicken.

#### Eingabefelder

*Remote Host* IP-Adresse der Station, in die Daten zu schreiben sind.

*Remote Port* Verbindungsadresse der fremden Station.

*Local Port* Verbindungsadresse der eigenen Station. Zur Vereinfachung können Sie für Remote und Local die gleiche Adresse verwenden.

*Time (10mSec)* Einstellbarer Zeitintervall für zyklisches Schreiben. Für das zyklische Schreiben muss der Timerwert mindestens 5 betragen.

#### Kontrollkästchen

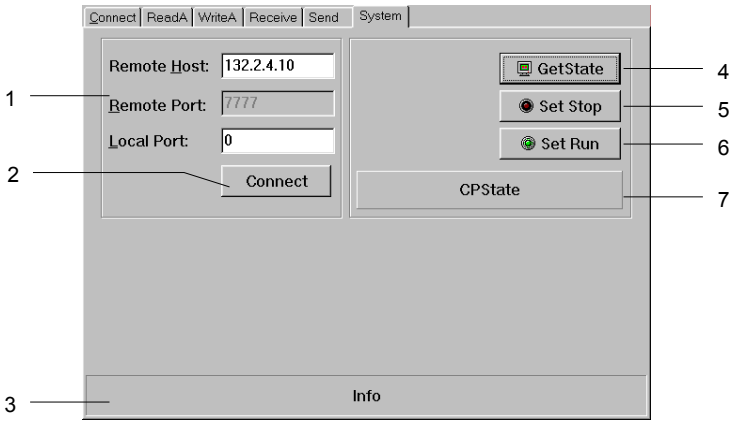
*UDP* Hier wählen Sie eine ungesicherte Kommunikation an. Bei der ungesicherten Kommunikation wird mit keiner virtuellen Verbindung gearbeitet. Das Telegramm wird als UDP-Telegramm gesendet.

#### Schaltflächen

*Connect* Die Verbindung wird aufgebaut und für den Schreibvorgang vorbereitet.

*Send this* Über die durch *Connect* aufgebaute Verbindung werden die im ASCII-Feld eingegebenen Daten in den CP geschrieben.

### System-Register




[1] Verbindungsdaten
[2] Verbindung aufbauen
[3] Informationsfenster für Verbindungsstatus
[4] CP Statusabfrage
[5] CP in STOP
[6] CP in RUN
[7] Statusanzeige, die mit GetState angefordert wird

In diesem Dialogfenster erhalten Sie Informationen über Ihren spezifizierten Host-CP.

**Eingabefelder**

*Remote Host* IP-Adresse der Station, in die Daten zu schreiben sind.

*Remote Port* Verbindungsadresse der fremden Station.

*Local Port* Verbindungsadresse der eigenen Station. Zur Vereinfachung können Sie für Remote und Local die gleiche Adresse verwenden.

**Schaltflächen**

*Connect* Die Verbindung wird aufgebaut und für die Kommunikation vorbereitet.

*GetState* Über die durch *Connect* aufgebaute Verbindung wird der Status des CPs übertragen und im Statusfenster angezeigt. Angezeigt werden kann:

-Hardware-Stop (Run/Stop-Schalter am CP steht in Stop-Stellung).
Der CP ist mit dem Testprogramm nicht fernbedienbar.

-Hardware-Run (Run/Stop-Schalter am CP steht in Run-Stellung).
Der CP ist mit dem Testprogramm fernbedienbar.

-Software-Stop (Run/Stop-Schalter am CP muss in Run-Stellung stehen).
Der CP wurde mit *SetStop* in Stop geschaltet.

-Software-Run (Run/Stop-Schalter am CP muss in Run-Stellung stehen).
Der CP wurde mit *SetRun* in Run geschaltet.

*SetStop* Der CP wird in Stop geschaltet. Diese Funktion ist nur möglich, wenn am CP der Run/Stop-Schalter in Run-Stellung steht.

*SetRun* Der CP wird in Run geschaltet. Auch diese Funktion ist nur möglich, wenn am CP der Run/Stop-Schalter in Run-Stellung steht.

# Teil 6 Einsatz CPU 24x DP

**Überblick**

Inhalt diese Kapitels ist der Einsatz der CPU 24x DP unter Profibus. Nach einer kurzen Einführung wird die Parametrierung und der Einsatz mit einem IM208-Master von VIPA gezeigt. Ein Kommunikationsbeispiel beendet dieses Kapitel.

**Inhalt**


| Thema | Seite |
|---|---|
| **Teil 6 Einsatz CPU 24x DP** | **6-1** |
| Grundlagen | 6-2 |
| Projektierung der CPU 24x DP | 6-6 |
| Default-Adressierung mit DB 1 ändern | 6-9 |
| Zugriff auf Parameterdaten | 6-12 |
| Parametrierung der direkt gesteckten E/A-Module | 6-13 |
| Diagnosefunktionen der CPU 24x DP | 6-14 |
| Statusmeldung intern an CPU | 6-17 |
| Aufbaurichtlinien | 6-19 |
| Inbetriebnahme | 6-24 |
| Einsatz der Diagnose-LEDs | 6-27 |
| Beispiel | 6-28 |

# Grundlagen

**Allgemein**

Profibus ist ein internationaler offener Feldbus-Standard für Gebäude-, Fertigungs- und Prozessautomatisierung. Profibus legt die technischen und funktionellen Merkmale eines seriellen Feldbus-Systems fest, mit dem verteilte digitale Feldautomatisierungsgeräte im unteren (Sensor-/Aktor-Ebene) bis mittleren Leistungsbereich (Prozessebene) vernetzt werden können.

Profibus besteht aus einem Sortiment kompatibler Varianten. Die hier angeführten Angaben beziehen sich auf den Profibus DP.

**Profibus-DP**

Profibus-DP ist besonders geeignet für die Fertigungsautomatisierung. DP ist sehr schnell, bietet Plug and Play und ist eine kostengünstige Alternative zur Parallelverkabelung zwischen SPS und dezentraler Peripherie. Profibus-DP ist für den schnellen Datenaustausch auf der Sensor-Aktor-Ebene konzipiert. Hier kommunizieren zentrale Steuergeräte wie SPS über eine schnelle, serielle Verbindung mit dezentralen Ein- und Ausgangsgeräten.

Der Datenaustausch erfolgt vorwiegend zyklisch. Die Datenübertragung basiert bei Profibus-DP auf einer hocheffizienten Telegrammstruktur. Während eines einzigen Buszyklus liest der Master die Eingangswerte der Slaves und schreibt die Ausgangsinformationen an die Slaves.

**Master und Slaves**

Profibus unterscheidet zwischen aktiven Stationen (Master) wie SPS oder PC und passiven Stationen (Slave).

*Master-Geräte*

Master-Geräte bestimmen den Datenverkehr auf dem Bus. Durch das Busprotokoll wird ein logischer Tokenring zwischen den intelligenten Geräten aufgebaut.

Ein Master (IM 208 DP bzw. IM 208 DPO) darf Nachrichten ohne externe Aufforderung aussenden, wenn er in Besitz der Buszugriffsberechtigung (Token) ist. Master werden im Profibus-Protokoll auch als aktive Teilnehmer bezeichnet.

*Slave-Geräte*

Slave-Geräte wie Peripheriegeräte, Sensoren, Aktoren, Messumformer und die VIPA Profibus-Koppler (IM 253 DP, IM 253 DPO und die CPU 24x DP) sind typische Slave-Geräte.

Diese Geräte haben gemäß der Profibus-Norm keine Buszugriffsberechtigung. Sie dürfen nur Nachrichten quittieren oder auf Anfrage eines Masters Nachrichten an diesen übermitteln. Slaves haben einen sehr geringen Anteil am Busprotokoll. Slaves werden auch als passive Teilnehmer bezeichnet.

## Kommunikation

Das Busübertragungsprotokoll bietet zwei Verfahren für den Buszugriff:

### Master mit Master

Die Master-Kommunikation wird auch als Token-Passing-Verfahren bezeichnet. Das Token-Passing-Verfahren garantiert die Zuteilung der Buszugriffsberechtigung. Das Zugriffsrecht auf den Bus wird zwischen den Geräten in Form eines "Token" weitergegeben. Der Token ist ein spezielles Telegramm, das über den Bus übertragen wird.

Wenn ein Master den Token besitzt, hat er das Buszugriffsrecht auf den Bus und kann mit allen anderen aktiven und passiven Geräten kommunizieren. Die Tokenhaltezeit wird bei der Systemkonfiguration bestimmt. Nachdem die Tokenhaltezeit abgelaufen ist, wird der Token zum nächsten Master weitergegeben, der dann den Buszugriff hat und mit allen anderen Geräten kommunizieren kann.

### Master-Slave-Verfahren

Der Datenverkehr zwischen dem Master und den ihm zugeordneten Slaves wird in einer festgelegten immer wiederkehrenden Reihenfolge automatisch durch den Master durchgeführt. Bei der Projektierung bestimmen Sie die Zugehörigkeit des Slaves zu einem bestimmten Master. Weiter können Sie definieren, welche DP-Slaves für den zyklischen Nutzdatenverkehr aufgenommen oder ausgenommen werden.

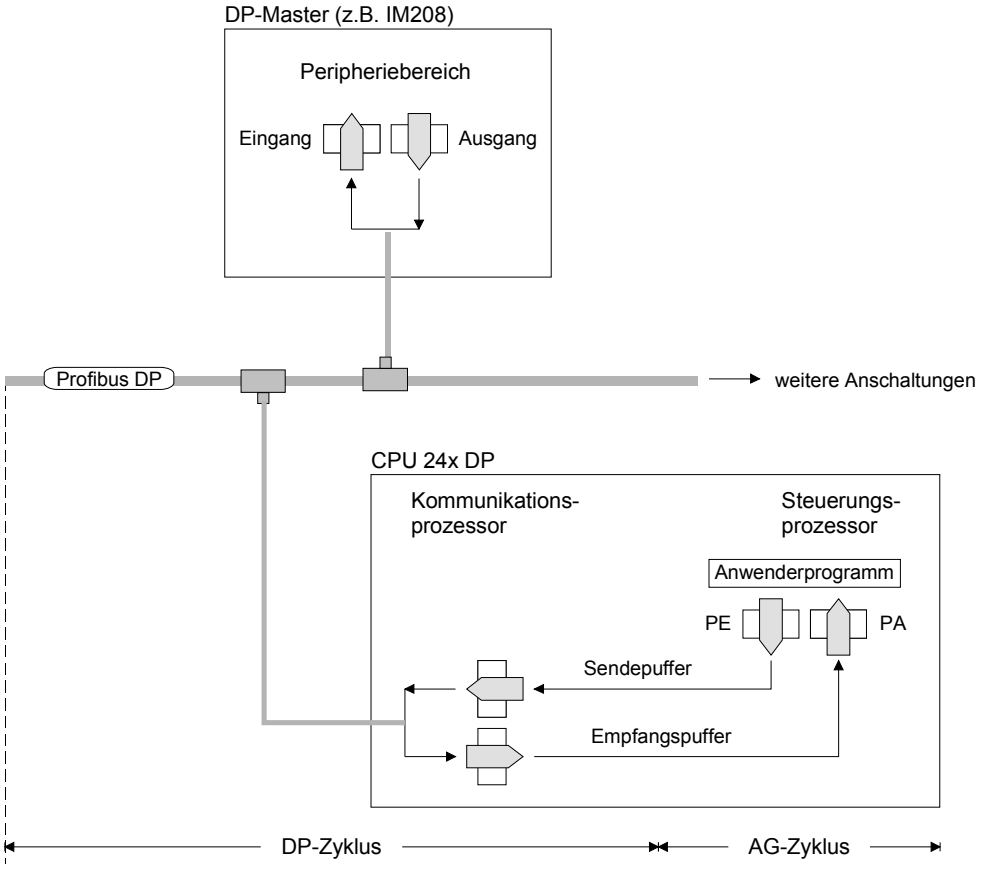
Der Datentransfer zwischen Master und Slave gliedert sich in Parametrierungs-, Konfigurierungs- und Datentransfer-Phasen. Bevor ein DP-Slave in die Datentransfer-Phase aufgenommen wird, prüft der Master in der Parametrierungs- und Konfigurationsphase, ob die projektierte Konfiguration mit der Ist-Konfiguration übereinstimmt. Überprüft werden Gerätetyp, Format- und Längeninformationen und die Anzahl der Ein- und Ausgänge. Sie erhalten so einen zuverlässigen Schutz gegen Parametrierfehler.

Zusätzlich zum Nutzdatentransfer den der Master selbständig durchführt, können Sie neue Parametrierdaten an einen Bus-Koppler schicken.

Mit diesen zwei Zugriffsmethoden können Sie Mono- oder Multi-Master-Systeme realisieren. Sie erreichen dadurch ein hohes Maß an Flexibilität bei der Systemkonfiguration.

**Funktionsweise der Daten-übertragung**

Der Datenaustausch zwischen DP-Master und CPU 24x DP erfolgt zyklisch über Sende- und Empfangspuffer.




PE: Peripheriebereich der Eingänge
PA: Peripheriebereich der Ausgänge

Die Datenübertragung läuft in zwei Zyklen ab; im AG-Zyklus und im DP-Zyklus.

**AG-Zyklus**

Die Sendedaten werden durch das Anwenderprogramm in den Peripheriebereich der Ausgänge (PA) der CPU geschrieben.

Am Zykluskontrollpunkt kopiert der Kommunikationsprozessor die Sendedaten aus dem PA in seinen DP - Sendepuffer. Gleichzeitig kopiert der Kommunikationsprozessor die Empfangsdaten in den Peripheriebereich der Eingänge (PE). Die im PE abgelegten Empfangsdaten können durch das Anwenderprogramm ausgewertet werden.

Der Datenaustausch zwischen Steuerungs- und Kommunikationsprozessor erfolgt am Zykluskontrollpunkt.

**DP-Zyklus**

Der DP-Master schickt an die CPU 24x DP Daten, die im Empfangspuffer des Kommunikationsprozessors abgelegt werden.

Gleichzeitig werden die Sendendaten der CPU 24x DP vom DP-Master abgeholt. Der Datenaustausch zwischen DP-Master und DP-Slave über den Bus erfolgt zyklisch, unabhängig vom Zykluskontrollpunkt der CPU 24x DP.

**Datenkonsistenz**

Daten bezeichnet man als konsistent, wenn sie inhaltlich zusammengehören. Inhaltlich gehören zusammen: das High- und Low-Byte eines Analogwerts (wortkonsistent) und das Kontroll- und Status-Byte mit zugehörigem Parameterwort für den Zugriff auf die Register.

Die Datenkonsistenz ist im Zusammenspiel von Peripherie und Steuerung grundsätzlich nur für 1 Byte sichergestellt. Das heißt, die Bits eines Bytes werden zusammen eingelesen bzw. ausgegeben. Für die Verarbeitung digitaler Signale ist eine byteweise Konsistenz ausreichend.

Für Daten, deren Länge ein Byte überschreiten, wie z.B. bei Analogwerten muss die Datenkonsistenz erweitert werden. Profibus garantiert die Konsistenz mit der erforderlichen Länge. Bitte beachten Sie, dass Sie die konsistenten Daten auf die richtige Art vom Profibus-Master in Ihre SPS übernehmen.

Hinweise hierzu finden Sie im Handbuch zu Ihrem Profibus-Master bzw. zur Anschaltbaugruppe.

**Einschränkungen**

Bei Ausfall eines übergeordneten Masters wird dies von der CPU nicht automatisch erkannt. Für diesen Fall sollten Sie immer ein Kontroll-Byte mitschicken, das die Präsenz des Masters mitteilt und somit gültige Masterdaten kennzeichnet.

Den Einsatz des Kontroll-Bytes finden Sie auch im Beispiel am Ende dieses Kapitels.

# Projektierung der CPU 24x DP

**Einbindung in Profibus**

Im Gegensatz zum VIPA Profibus-Slave IM 253 DP, ist der Profibus-Koppler in der CPU 24x DP ein "intelligenter Koppler".

Der "Intelligente Koppler" verarbeitet Daten, die in einem Ein- bzw. Ausgabe-Bereich der CPU stehen. Für Ein- bzw. Ausgabe-Daten werden getrennte Speicher-Bereiche genutzt.

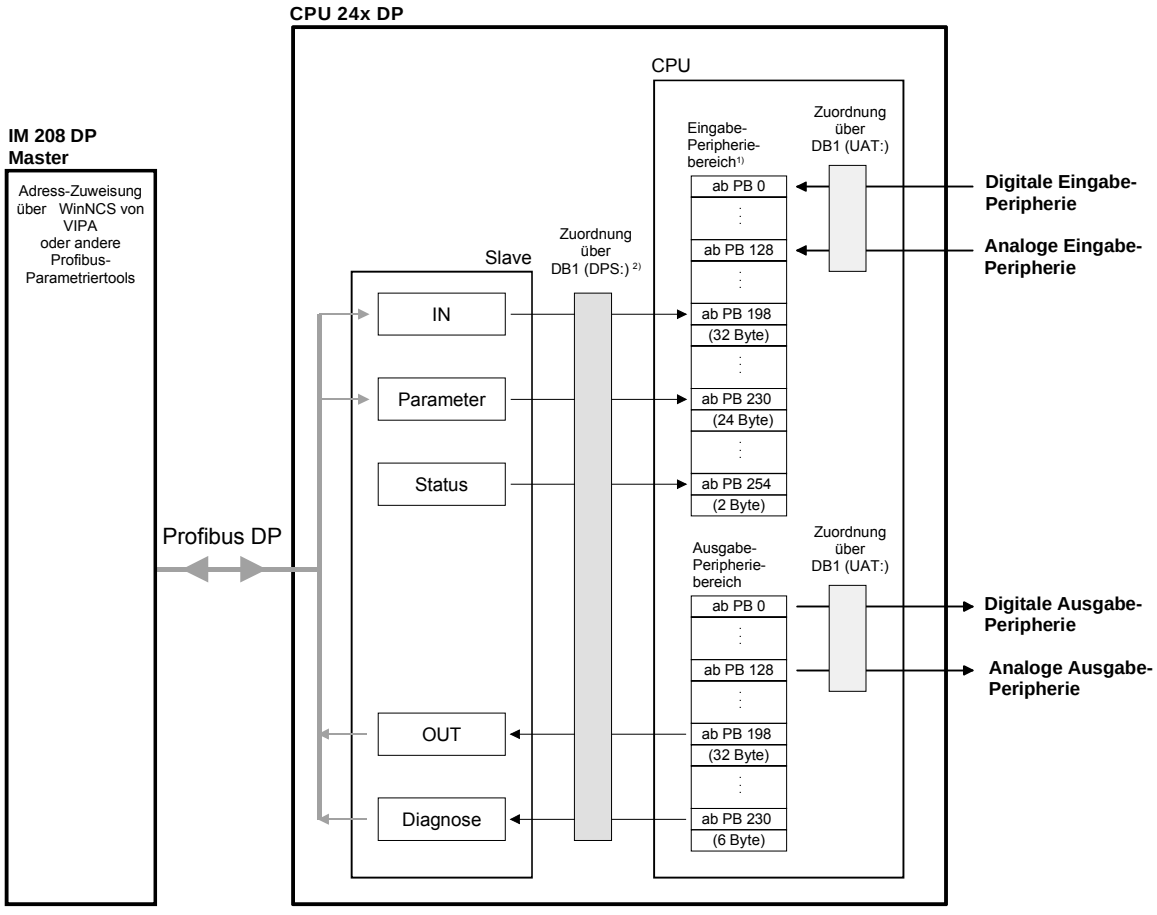
Die Adresszuordnung erfolgt automatisch mit Default-Werten (s. Abb.). Sie kann aber jederzeit durch Programmierung des DB1 (Label DPS:) verändert werden.

Beachten Sie hierbei, dass es nicht zu Adressüberschneidungen kommt.

Neben den Profibus-Daten werden auch die direkt gesteckten Module in den Peripherie-Adressraum eingebunden.

Digitale und analoge Module haben getrennt Adressbereiche. Digital-Module liegen im Adress-Bereich 0 ... 127 und Analog-Module im Bereich 128 ... 255. Auch hier können Sie die Adresszuordnung jederzeit durch Programmierung des DB 1 (Label UAT:) ändern.

Zur Einbindung der direkt gesteckten Module in den Profibus, sind mit einem Programm deren Daten innerhalb der CPU in den Profibus-Adress-Bereich zu transferieren.




1) die hier eingestellten Werte sind Defaulteinstellungen
2) bei Defaultparametrierung DPS im DB1 nicht erforderlich

**direkt gesteckten Master einbinden**

Befindet sich zusätzlich zur CPU 24x DP ein Master am gleichen Rückwandbus, so werden beim Hochlauf der CPU die Master-Parameter ermittelt und der Master im Peripherie-Adress-Bereich eingebunden.

Den Adress-Bereich, der in der CPU belegt wird, geben Sie mit Ihrem Profibus-Master-Parametrier-Tool vor.

**Achtung!**

Bitte beachten Sie, dass bei einer Adressüberschneidung die CPU in STOP geht!

**Übersicht**

Nachfolgend eine Übersicht, Zuordnung Peripherie-Master-Slave

**1** **Profibus Master IM 208**
Parametrierung über WinNCS
GSD-Datei erforderlich

**direkt gesteckte E-/A-Peripherie**
*Adressierung:*
automatisch oder über DB1 (Label UAT:) einbinden
*Parametrierung:*
über DB1 mit Label Pxx:

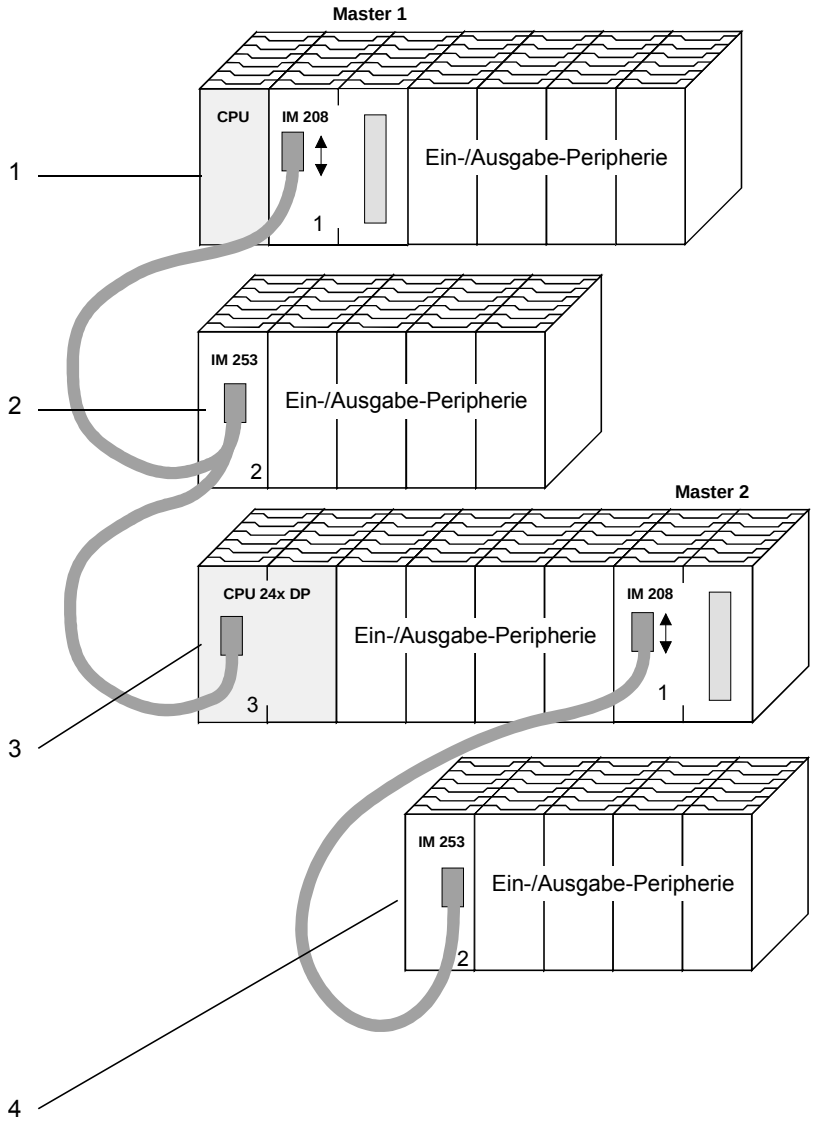
**2** **Profibus-Slave IM 253**
Parametrierung in Master 1 über WinNCS inkl. aller Peripheriemodule
GSD-Datei erforderlich

**3** **CPU 24x DP**
*Einblendung DP-Slave in CPU*
Automatisch oder über DB1 mit Label DPS:
*DP-Slave*
Einbindung in Profibus in Master 1 über WinNCS ohne E/A-Peripherie
GSD-Datei erforderlich

**direkt gesteckte E-/A-Peripherie**
*Adressierung:*
automatisch oder über DB1 mit Label UAT:
*Parametrierung:*
über DB1 mit Label Pxx:

**Profibus Master IM 208**
Parametrierung über WinNCS
GSD-Datei erforderlich
*Adressierung*
Automatisch oder über DB1 mit Label UAT:

**4** **Profibus-Slave IM 253**
Parametrierung in Master 2 über WinNCS inkl. aller Peripheriemodule
GSD-Datei erforderlich




Den DB1 können Sie auch komfortabel unter WinNCS erzeugen und als s5d-Datei in Ihre CPU übertragen! Hierzu finden Sie ein Beispiel am Ende des Kapitels.

**Projektierung in übergeordnetem Master**

Für die Einbindung der CPU 24x DP in ein übergeordnetes Mastersystem ist die mitgelieferte GSD-Datei erforderlich.

**GSD-Datei**

Geeignete Master-Anschaltungen für die SPS sind von VIPA die IM 208 DP und IM 208 DPO und die IM 308-C von Siemens. Zur Konfiguration einer Slave-Anschaltung bekommen Sie die Leistungsmerkmale der VIPA CPU 24x DP in Form einer GSD-Datei mitgeliefert. Aufbau, Inhalt und Codierung der GSD-Datei sind durch die Norm festgelegt. Die GSD ermöglicht die komfortable Projektierung beliebiger DP-Slaves mit Projektierungsgeräten verschiedener Hersteller. Diese Daten werden bei der Profibus Nutzerorganisation (PNO) herstellerübergreifend archiviert und können jederzeit abgerufen werden.

**Die GSD-Datei für die CPU 24x DP von VIPA lautet: VIPA04D5.GSD**

Installieren Sie diese GSD-Datei in Ihrem Projektiertool. Nähere Hinweise zur Installation der GSD-Datei finden Sie im Handbuch zu Ihrem Projektiertool.

In der GSD-Datei ist z.B. auch die Ident-Nummer abgelegt unter der die Busgeräte spezifiziert sind. Die Ident-Nummern werden zentral von der PNO verwaltet und lauten für VIPA-Bus-Geräte:

**Ident-Nummer CPU 24x DP von VIPA: 04D5h**

**GSD-Datei in WinNCS einbinden**

WinNCS ist das VIPA-Projektier-Tool für die Profibus-Master-Baugruppen von VIPA. In der Regel wird WinNCS immer mit der neuesten GSD-Datei ausgeliefert. Sie können aber jederzeit in WinNCS andere GSD-Dateien einbinden.

Gehen Sie hierzu in das Verzeichnis WinNCS, das bei der Installation angelegt wird und kopieren Sie Ihre GSD-Datei in das Verzeichnis GSD.

Beim nächsten Start von WinNCS wird die neue GSD-Datei verwendet.

# Default-Adressierung mit DB 1 ändern

Beim Hochlauf der CPU werden die am Rückwand-Bus befindlichen Module ermittelt und ihnen Adressen im Peripheriebereich der CPU zugeordnet.

Hierbei ordnet die CPU 24x DP den Ein und Ausgabedaten des integrierten Profibus-Kopplers Speicherbereiche zu. Die Zuordnung erfolgt entweder automatisch über Default-Adressen oder manuell durch Programmierung des DB 1. Über Lese- bzw. Schreibzugriffe auf diese Peripheriebytes können Sie die Module, die am Profibus hängen, ansprechen.

Den DB 1 können Sie auch komfortabel unter WinNCS erzeugen. Hierzu finden Sie ein umfassendes Beispiel am Ende des Kapitels.

**Adress-Bereiche für Profibus-Slave**

Folgende Bereiche sind in der CPU für den Profibus-Slave reserviert:

| Datentyp | Datenbreite | Peripheriebereich (default) |
|---|---|---|
| Eingangsdaten IN | max. 64 Bytes | 32 Bytes ab PW 198 |
| Ausgangsdaten OUT | max. 64 Bytes | 32 Bytes ab PW 198 |
| Parameterdaten IN | 24 Bytes | 24 Bytes ab PW 230 |
| Diagnosedaten OUT | 6 Bytes | 6 Bytes ab PW 230 |
| Statusdaten IN | 2 Bytes | 2 Bytes ab PW 254 |

**Label DPS**

Mit dem Label DPS (DP-Slave) können Sie Datenbereiche für den integrierten Profibus-Koppler neu vergeben. Zu diesem Label gehören immer 7Byte Parameter, die nach dem Label "DPS:" aufzuführen und je mit einem Leerzeichen voneinander zu trennen sind.

Die Bytes haben folgende Belegung:

| Byte | Funktion | Wertebereich |
|---|---|---|
| 0 | Startadresse Eingangsdaten | 0 ... 255 |
| 1 | Datenbreite Eingangsdaten | 0 ... 64 |
| 2 | Startadresse Ausgangsdaten | 0 ... 255 |
| 3 | Datenbreite Ausgangsdaten | 0 ... 64 |
| 4 | Startadresse Parameterdaten | 0 ... 232*) |
| 5 | Startadresse Diagnosedaten | 0 ... 250*) |
| 6 | Startadresse Statusdaten | 0 ... 254*) |

*) Durch Eingabe von 255 werden diese Daten ausgeblendet d.h. Speicherplatz wird für diese Daten nicht belegt.

**Hinweis!**

Bei Einsatz des Labels DPS sind immer die 7Byte Parameter anzugeben!

**Beispiel**

Das folgende Beispiel zeigt den Einsatz des Labels DPS im DB 1. Folgende Werte sollen über den DB 1 eingestellt werden:

- Eingangsdaten ab Adresse 100 mit einer Breite von 16 Byte
- Ausgangsdaten ab Adresse 0 mit einer Breite von 64 Byte
- Keine Parameterdaten
- Diagnosedaten ab Adresse 230 (feste Breite 6 Byte)
- Statusdaten ab Adresse 240 (feste Breite 2 Byte)

Hiermit ergeben sich folgende Daten für den DB 1:

```
DB1
   KC= DPS: 100 16 0 64 255 230 240 ;
```

**Hinweis!**

Zwischen jedem Label, jeder Zahl und jedem Semikolon muss ein Leerzeichen stehen! (siehe auch Beispiel am Ende des Kapitels)

**Adress-Bereiche für direkt gesteckte E/A-Module**

Folgende Anfangs-Bereiche sind in der CPU für die direkt gesteckten E/A-Module reserviert:

| Modultyp | Peripheriebereich |
|---|---|
| digitales Eingabe-Modul | ab PB0 im Eingabe-Bereich |
| analoges Eingabe-Modul | ab PB128 im Eingabe-Bereich |
| digitales Ausgabe-Modul | ab PB0 im Ausgabe-Bereich |
| analoges Ausgabe-Modul | ab PB128 im Ausgabe-Bereich |

**Label UAT**

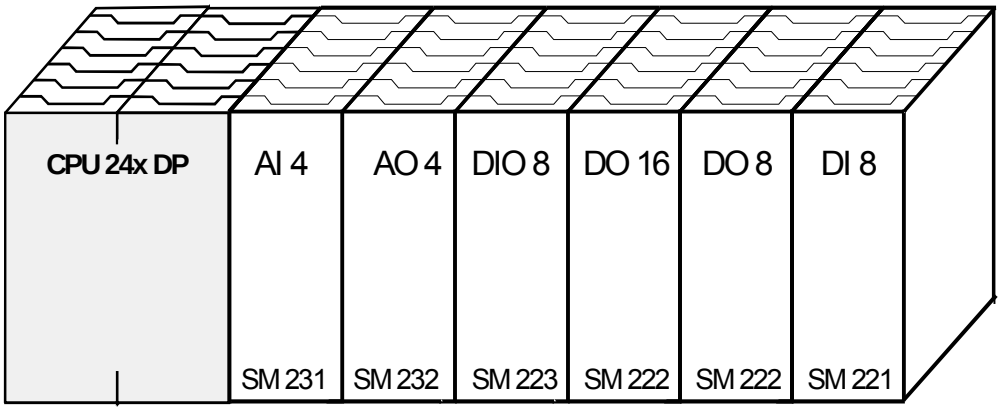
Mit dem Label UAT (User Adress Table) können Sie Datenbereiche für die direkt gesteckten E/A-Module neu vergeben, indem Sie hinter dem Label UAT die Peripherieadresse entsprechend der gesteckten Reihenfolge eingeben. Bei der Programmierung des DB 1 ist folgendes zu beachten:

- Jede Adresse besteht aus 2Byte, wobei das High-Byte immer 0 sein muss.
- Bei kombinierten Modulen ist zuerst die Eingabe- und dann die Ausgabe-Adresse zu parametrieren.
- Analog-Module können ab Adresse 0 abgelegt werden. Sie liegen dann im Prozessabbild und erfahren in jedem Zyklus ein Update. Bitte beachten Sie, dass sich hierdurch die Zykluszeit verlängert.
- Die Eingabe erfolgt im String-Format (KC-Format).

**Achtung!**
Bitte beachten Sie, dass bei einer Adressüberschneidung im DB 1 die CPU in STOP geht.

Das folgende Beispiel soll die manuelle Vergabe von Peripherieadressen verdeutlichen:

**Beispiel**
Ihr System habe folgenden Aufbau:




Für diesen Aufbau möchten Sie z.B. folgende Adressen parametrieren:

| Modul<br>Speicherbedarf | AI 4<br>8Byte | AO 4<br>8Byte | DIO 8<br>je 1Byte | DO 16<br>2Byte | DO 8<br>1Byte | DI 8<br>1Byte |
|---|---|---|---|---|---|---|
| Eingabe-Adresse | 160 | | 24 | | | 25 |
| Ausgabe-Adresse | | 160 | 24 | 10 | 12 | |

Ergänzen Sie den DB1 um das Label UAT, gefolgt von der Adressierung im KC-Format:

```
KC= UAT: 0 160 0 160 0 24 0 24 0 10 0 12 0 25 ;
```

Beim Hochlauf der CPU wird der DB 1 erkannt und die Adressen werden entsprechend zugeteilt.

# Zugriff auf Parameterdaten

Die Parameterdaten beinhalten die Parametrierung eines Profibus-Slaves. Diese Daten werden bei der Masterprojektierung erzeugt.

Die Parameterdaten bestehen aus profibusspezifischen Daten (Busparameter) und benutzerspezifischen Daten in denen bei der CPU 24x DP die Ein- und Ausgabe-Bytes definiert sind.

Die benutzerspezifischen Daten (Byte 7 ... 9) werden in das Prozessabbild eingeblendet.

**Aktualisierung**

Der Master schickt neue Parameterdaten wenn

- die CPU 24x DP sich im Hochlauf befindet
- die Verbindung zwischen CPU 24x DP und Master gestört war wie z.B. kurzzeitiges Abziehen des Bus-Steckers.

**Aufbau**

Das Parameterdaten-Telegramm ist 32Byte lang und hat folgenden prinzipiellen Aufbau:

| *Byte* | *Bezeichnung* | *Beschreibung* |
|---|---|---|
| 0 ... 6 | Profibus-Norm-Parameterdaten | Die Normparameter können nicht ausgegeben werden. |

Die nachfolgenden 24Byte werden im Prozessabbild eingeblendet.

| | | |
|---|---|---|
| 7 ... 9 | Spec_User_Prm_Byte | Dieser Bereich ist für zukünftige Erweiterungen reserviert. Hier wird immer 0 eingetragen. |
| 10 ... 31 | User_Prm_Data | Hier finden Sie die Konfigurationsbytes Ihrer Ein-/Ausgabe-Peripherie, die Sie unter WinNCS parametriert haben. |

# Parametrierung der direkt gesteckten E/A-Module

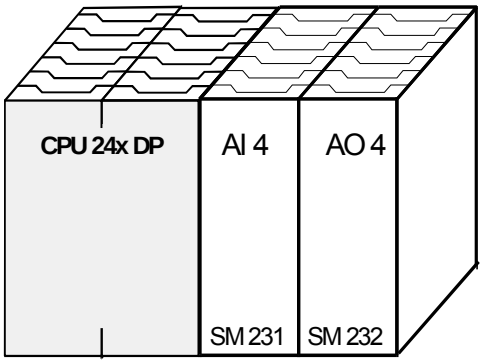
Direkt gesteckte Module wie z.B. Analog-Module können bis zu 16Byte-Parametrierdaten von der CPU erhalten. Im DB 1 können Sie für jeden Steckplatz eine Parametrierung im Hex-Format angeben. Jedem Steckplatz ist ein Label zugeordnet:

Steckplatz (1...32) → Label (P01...P32)

Das folgende Beispiel soll die Parametrierung anhand von zwei Analog-Modulen verdeutlichen:

Beispiel

Ihr System habe folgenden Aufbau:




**AI 4x16Bit Multiinput**
- kein Diagnosealarm
- Kanal 0: Spannungsmessung (-10 ... +10V)
- Kanal 1: Strommessung (4 ... 20mA)
- Kanal 2: Pt100 (-200 ... 500°C)
- Kanal 3: keine Messung
- Defaultoptionen

**AO 4x12Bit Multioutput**
- kein Diagnosealarm
- Kanal 0: Stromausgabe (4 ... 20mA)
- Kanal 1: Spannungsausgabe (1 ... 5V)
- Kanal 2: keine Ausgabe

Hieraus ergibt sich für die zwei Analog-Module folgende Parametrierung:

**Parameterbereiche :**

**AI 4x16Bit Multiinput**

| | | |
|---|---|---|
| Byte 0 | 00h | Diagnosealarm-Byte |
| Byte 1 | 00h | reserviert |
| Byte 2 | 2Bh | Funktions-Nr. Kanal 0 |
| Byte 3 | 2Dh | Funktions-Nr. Kanal 1 |
| Byte 4 | 02h | Funktions-Nr. Kanal 2 |
| Byte 5 | 00h | Funktions-Nr. Kanal 3 |
| Byte 6 | 00h | Option-Byte Kanal 0 |
| Byte 7 | 00h | Option-Byte Kanal 1 |
| Byte 8 | 00h | Option-Byte Kanal 2 |
| Byte 9 | 00h | Option-Byte Kanal 3 |

**AO 4x12Bit Multioutput**

| | | |
|---|---|---|
| Byte 0 | 00h | Diagnosealarm-Byte |
| Byte 1 | 00h | reserviert |
| Byte 2 | 04h | Funktions-Byte Kanal 0 |
| Byte 3 | 02h | Funktions-Byte Kanal 1 |
| Byte 4 | 00h | Funktions-Byte Kanal 2 |
| Byte 5 | 01h | Funktions-Byte Kanal 3 |

Ergänzen Sie im KC-Format (String) den DB1 um den Eintrag Pxx (xx für den Steckplatz) gefolgt von den Parameterbytes in Hex. Trennen Sie die Einträge durch ein Semikolon:

```
KC= P01: 00 00 2B 2D 02 00 00 00 00 00 ;
    P02: 00 00 04 02 00 01 ;
```

Zwischen jedem Label, jeder Zahl und jedem Semikolon muss ein Leerzeichen stehen!

# Diagnosefunktionen der CPU 24x DP

**Übersicht**

Die umfangreichen Diagnosefunktionen von Profibus-DP ermöglichen eine schnelle Fehlerlokalisierung. Die Diagnosemeldungen werden über den Bus übertragen und beim Master zusammengefasst.

Die CPU 24x DP sendet auf Anforderung vom Master oder im Fehlerfall Diagnosedaten. Da ein Teil der Diagnosedaten (Byte 11 ... 15) im Peripherieadressbereich der CPU liegt, können Sie eine Diagnose auslösen und Diagnosedaten beeinflussen. Die Diagnosedaten bestehen aus:

- Norm-Diagnose-Daten (Byte 0 ... 5),
- Gerätebezogene Diagnose-Daten (Byte 6 ... 15).

**Aufbau**

Die Diagnosedaten haben folgenden Aufbau:

*Norm-Diagnosedaten*

| | |
|---|---|
| Byte 0 | Stationsstatus 1 |
| Byte 1 | Stationsstatus 2 |
| Byte 2 | Stationsstatus 3 |
| Byte 3 | Master-Adresse |
| Byte 4 | Ident-Nummer (low) |
| Byte 5 | Ident-Nummer (high) |

*Gerätebezogene Diagnosedaten*

| | |
|---|---|
| Byte 6 | Länge und Code gerätebezogene Diagnose |
| Byte 7 | Gerätebezogene Diagnosemeldungen |
| Byte 8 ... Byte 10 | reserviert |
| Byte 11 ... Byte 15 | Anwenderspezifische Diagnosedaten werden in CPU-Peripherieadress-bereich eingeblendet und können bearbeitet werden. |

**Auslösen einer Diagnose**

Byte 11 beinhaltet das Diagnose-Statusbyte hier können Sie eine Diagnose auslösen. Näheres hierzu finden Sie im Hinweis bei der gerätebezogenen Diagnose auf den Folgeseiten.

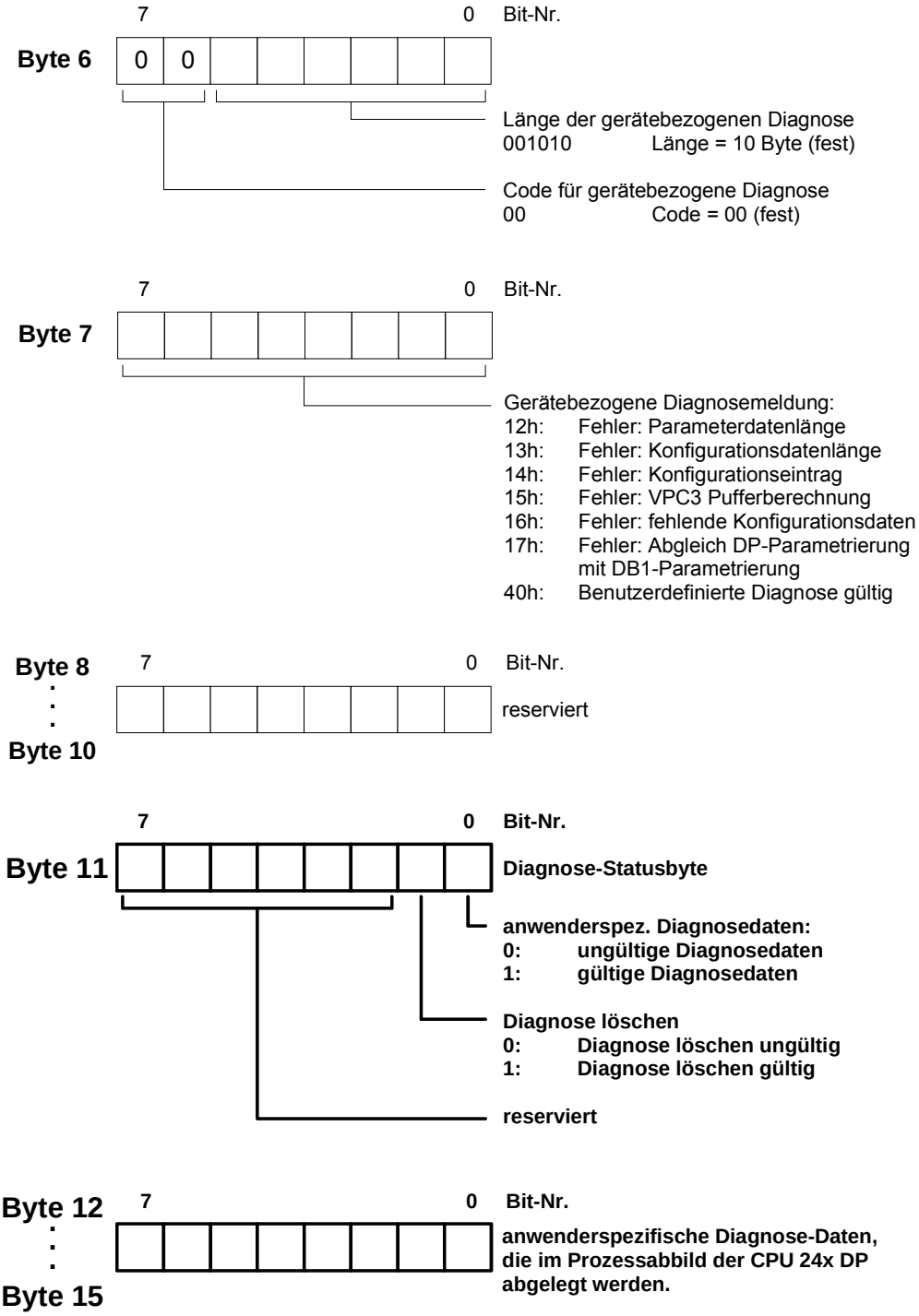
**Norm-Diagnosedaten**

Nähere Angaben zum Aufbau der Norm-Diagnosedaten finden Sie in den Profibus-Norm-Schriften. Die Normschriften sind bei der Profibus Nutzer Organisation erhältlich.

**gerätebezogene Diagnosedaten**

Die gerätebezogenen Diagnosedaten geben detaillierte Auskunft über den Slave und die Peripherie-Module. Die Länge der gerätebezogenen Diagnosedaten ist fest auf 10Byte eingestellt.

Die Byte 11 ...15 liegen im Prozessabbild ab.

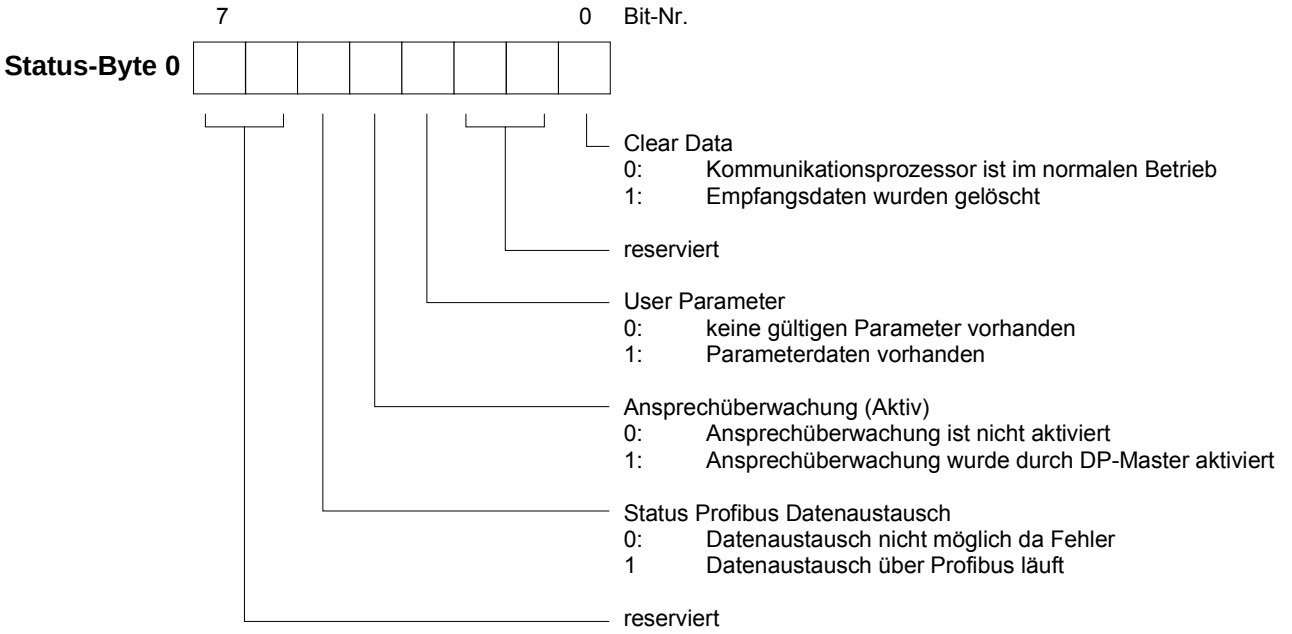


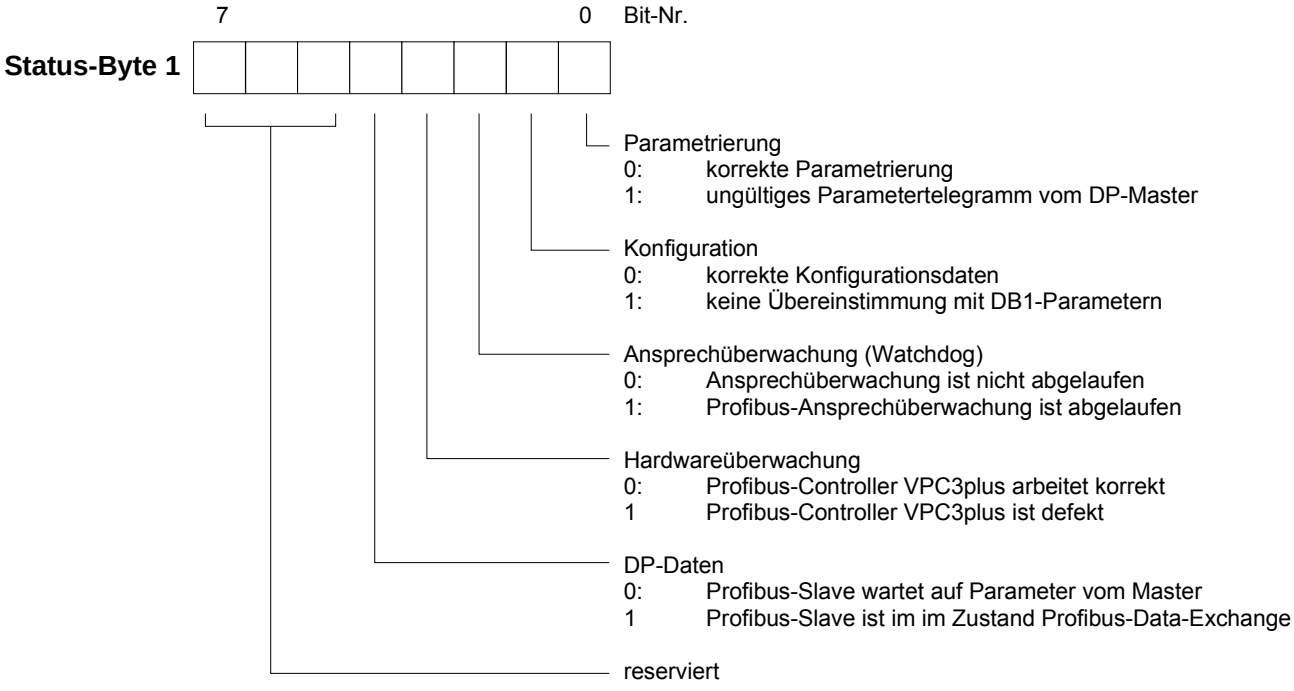

**Hinweis!**

Durch Zustandswechsel von 0 → 1 im Diagnose-Statusbyte lösen Sie eine Diagnose aus und das entsprechende Diagnose-Telegramm wird übertragen. **Der Zustand 0000 0011 wird ignoriert!**

# Statusmeldung intern an CPU

Den aktuellen Status der Profibus-Kommunikation finden Sie in den Statusmeldungen, die in den Peripherieadressbereich der CPU eingebunden sind. Die Statusmeldungen bestehen aus 2Byte und haben folgenden Aufbau:



## Parameter

| | |
|---|---|
| **Clear Data** | Im Fehlerfall werden die Sende- und Empfangspuffer gelöscht. |
| **reserviert** | Diese zwei Bits sind für zukünftige Erweiterungen reserviert. |
| **User Parameter** | Zeigt die Gültigkeit der Parameterdaten an. Die Parameterdaten werden im Master-Parametriertool eingegeben. |
| **Ansprechüber-wachung (Aktiv)** | Zeigt den Zustand der Aktivierung der Ansprechüberwachung im übergeordneten Profibus-Master an. Bei überschrittener Ansprechüberwachungszeit bricht der Slave die Kommunikation ab. |
| **Status Profibus Datenaustausch** | Statusanzeige über die Kommunikation mit dem übergeordneten Master. Bei fehlerhafter Konfiguration oder bei fehlerhaften Parametern wird die Kommunikation unterbrochen und der Fehler über dieses Bit angezeigt. |
| **Parametrierung** | Zeigt den Status der Parametrierdaten an. Die Länge der Parametrierdaten und die Anzahl der Parametrier-Bytes wird ausgewertet. Nur wenn diese gleich sind und nicht mehr als 31Byte Parameterdaten übertragen werden, ist die Parametrierung korrekt. |
| **Konfiguration** | Statusanzeige der Konfigurationsdaten, die vom Profibus-Master geschickt werden. Die Konfiguration erstellen Sie im Master Projektier-Tool. |
| **Ansprechüber-wachung (Watchdog)** | Hier wird der Zustand der Ansprechüberwachung im Profibus-Master angezeigt. Bei aktivierter Ansprechüberwachung und überschrittener Ansprechzeit im Slave wird hier ein Fehler angezeigt. |
| **Hardwareüber-wachung** | Ein gesetztes Bit zeigt hier an, der Profibus-Controller VPC3plus in der CPU 24x DP defekt ist. Kontaktieren Sie in diesem Fall die VIPA Hotline. |
| **DP-Daten** | Bei jedem Transferfehler über Profibus wird dieses Bit gesetzt. |

# Aufbaurichtlinien

**Profibus allgemein**

- Ein VIPA Profibus-DP-Netz darf nur in Linienstruktur aufgebaut werden.
- Profibus DP besteht aus mindestens einem Segment mit mindestens einem Master und einem Slave.
- Ein Master ist immer in Verbindung mit einer CPU einzusetzen.
- Profibus unterstützt max. 125 Teilnehmer.
- Pro Segment sind max. 32 Teilnehmer zulässig.
- Die maximale Segmentlänge hängt von der Übertragungsrate ab:

| | | |
|---|---|---|
| 9,6 ... 187,5kBaud | → | 1000m |
| 500kBaud | → | 400m |
| 1,5MBaud | → | 200m |
| 3 ... 12MBaud | → | 100m |

- Maximal 10 Segmente dürfen gebildet werden. Die Segmente werden über Repeater verbunden. Jeder Repeater zählt als Teilnehmer.
- Alle Teilnehmer kommunizieren mit der gleichen Baudrate. Die Slaves passen sich automatisch an die Baudrate an.

**optisches System**

- Es darf nur ein optischer Master in einer Linie verwendet werden.
- Mehrere Master dürfen mit einer CPU eingesetzt werden sofern sich diese auf dem gleichen Rückwandbus befinden (max. Stromaufnahme beachten).
- Die maximale LWL-Länge darf zwischen zwei Slaves bei 12MBaud maximal 50m betragen.
- Der Bus muss nicht abgeschlossen werden.

**Hinweis!**

Sie sollten bei optischen Teilnehmer am Busende die Buchse für den nachfolgenden Teilnehmer abdecken ansonsten besteht Blendungsgefahr und das Empfangsteil kann durch Fremdeinstahlung gestört werden. Verwenden Sie hierzu die mitgelieferten Gummi-Stäbchen und stecken Sie die Stäbchen in die übrigen zwei Öffnungen des LWL-Anschlusses.

**elektrisches System**

- Der Bus ist an beiden Enden abzuschließen.
- Master und Slaves sind beliebig mischbar.

**gemischtes System**

- Ein LWL-Master darf nur direkt über einen Optical Link Plug in ein elektrisches System eingekoppelt werden, d.h. zwischen Master und OLP darf sich kein Slave befinden.
- Zwischen zwei Mastern darf sich maximal eine Umsetzung (OLP) befinden.

**Aufbau und Einbindung in Profibus**

- Bauen Sie Ihr Profibus-System mit den entsprechenden Modulen auf.
- Stellen Sie an Ihren Buskoppler eine Adresse ein, die in Ihrem Bus noch nicht verwendet wird.
- Übertragen Sie die mitgelieferte GSD-Datei in Ihr Projektiersystem und projektieren Sie Ihr System.
- Übertragen Sie Ihre Projektierung in Ihren Master.
- Schließen Sie das Profibus-Kabel am Koppler an und schalten Sie die Spannungsversorgung ein.

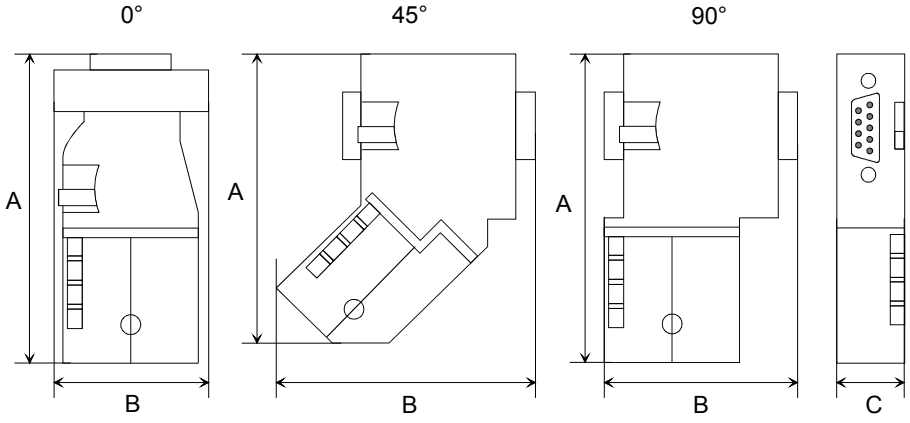
**Hinweis!**

Die Profibus-Leitung muss mit Ihrem Wellenwiderstand abgeschlossen werden. Bitte beachten Sie, dass Sie bei dem jeweiligen letzten Teilnehmer den Bus durch Zuschalten eines Abschlusswiderstands abschließen.

EasyConn Busanschluss-Stecker

In Systemen mit mehr als zwei Stationen werden alle Teilnehmer parallel verdrahtet. Hierzu ist das Buskabel unterbrechungsfrei durchzuschleifen.

Unter der Best.-Nr. VIPA 972-0DP10 erhalten Sie von VIPA den Stecker "EasyConn". Dies ist ein Busanschlussstecker mit zuschaltbarem Abschlusswiderstand und integrierter Busdiagnose.

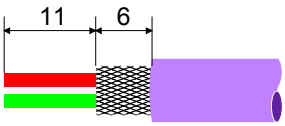


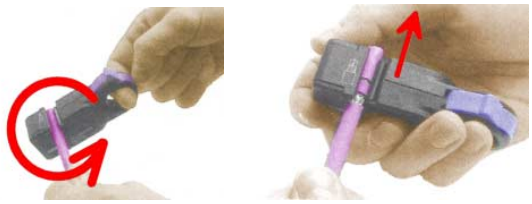

| | 0° | 45° | 90° |
|---|---|---|---|
| A | 64 | 61 | 66 |
| B | 34 | 53 | 40 |
| C | 15,8 | 15,8 | 15,8 |

Maße in mm

**Hinweis!**

Zum Anschluss des EasyConn-Steckers verwenden Sie bitte die Standard Profibus-Leitung Typ A (EN50170). Ab Ausgabestand 5 können auch hochflexible Bus-Kabel verwendet werden:
Lapp Kabel Best.-Nr.: 2170222, 2170822, 2170322.
Von VIPA erhalten Sie unter der Best.-Nr. VIPA 905-6AA00 das "EasyStrip" Abisolierwerkzeug, das Ihnen den Anschluss des EasyConn-Steckers sehr vereinfacht.

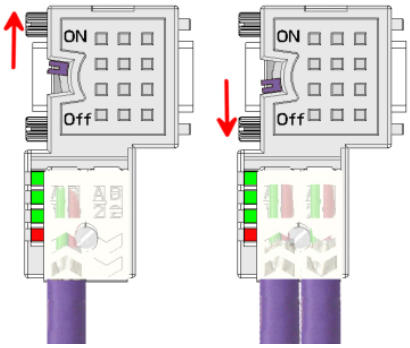


Maße in mm

Leitungsabschluss mit "EasyConn"

Auf dem "EasyConn" Busanschlussstecker von VIPA befindet sich unter anderem ein Schalter, mit dem Sie einen Abschlusswiderstand zuschalten können.



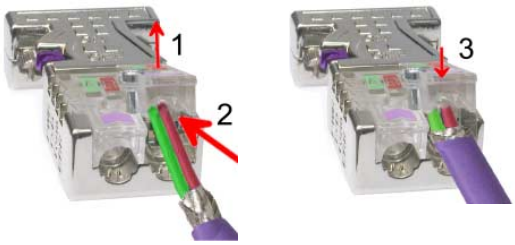

**Achtung!**

Der Abschlusswiderstand wird nur wirksam, wenn der Stecker an einem Slave gesteckt ist und der Slave mit Spannung versorgt wird.

**Hinweis!**

Eine ausführliche Beschreibung zum Anschluss und zum Einsatz der Abschlusswiderstände liegt dem Stecker bei.

Montage



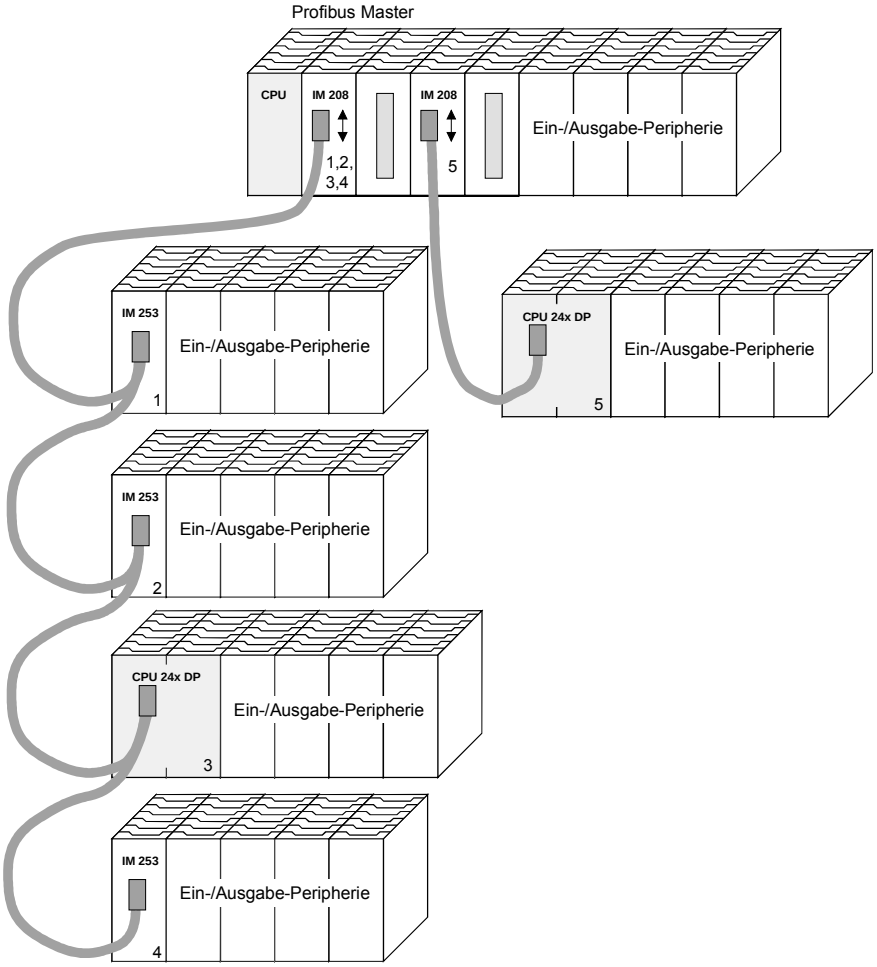

- Lösen Sie die Schraube.
- Klappen Sie den Kontaktdeckel auf.
- Stecken Sie beide Adern in die dafür vorgesehenen Öffnungen (Farbzuordnung wie unten beachten!).
- Bitte beachten Sie, dass zwischen Schirm und Datenleitungen kein Kurzschluss entsteht!
- Schließen Sie den Kontaktdeckel.
- Ziehen Sie die Schraube wieder fest (max. Anzugsmoment 4Nm).

**Bitte beachten:** Den grünen Draht immer an A, den roten immer an B anschließen!

**Beispiele für Profibus-Netze**

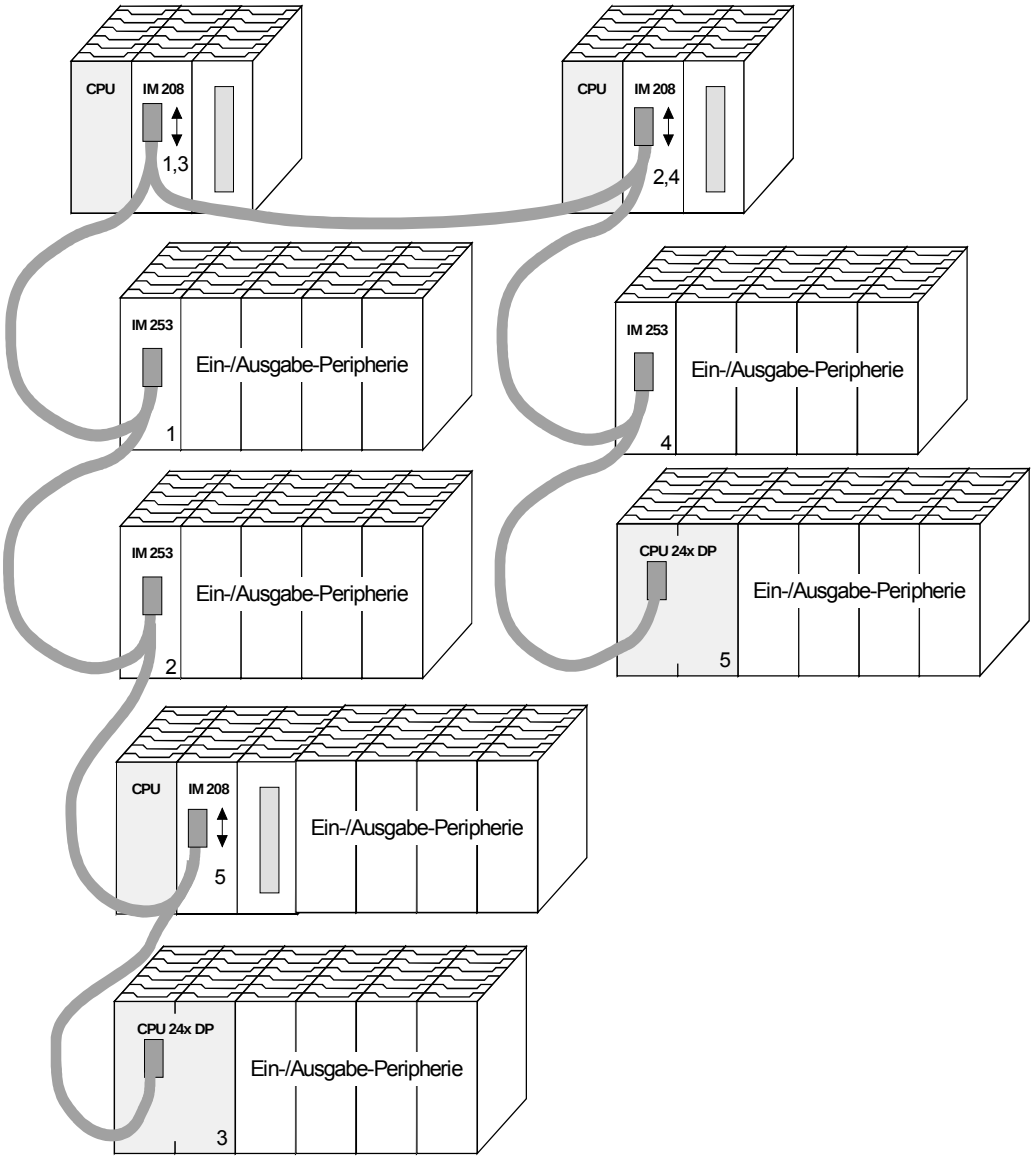
### Eine CPU und mehreren Master-Anschaltungen

Die CPU sollte eine kurze Zykluszeit haben, so ist gewährleistet dass die Daten von Slave Nr. 5 (rechts) immer aktuell sind. Dieser Aufbau ist nur sinnvoll, wenn am langsamen Strang (links) Slaves angekoppelt sind, deren Daten-Aktualität unwichtig ist. Hier sollten auch keine Module liegen, die einen Alarm auslösen.



**Multi Master System**

Mehrere Master-Anschaltungen an einem Bus zusammen mit mehreren Slaves:



# Inbetriebnahme

**Übersicht**

- Bauen Sie Ihre CPU 24x DP auf.
- Projektieren Sie die CPU 24x DP in Ihrem Mastersystem.
- Projektieren Sie die über den Rückwandbus verbundene E/A-Peripherie.
- Verbinden Sie Ihre CPU 24x DP mit Ihrem Profibus.
- Schalten Sie die Spannungsversorgung ein.

**Aufbau**

Bauen Sie Ihre CPU 24x DP mit den gewünschten Peripherie-Modulen auf. Bitte beachten Sie hierbei die maximale Strombelastung Ihrer Spannungsversorgung.

**Projektierung im Mastersystem**

Projektieren Sie Ihre CPU 24x DP in Ihrem Master-System. Für die Projektierung können Sie WinNCS von VIPA einsetzen.

**Hinweis!**

Direkt über den Rückwandbus verbundene E/A-Module können im Master nicht parametriert werden! Der Master verarbeitet lediglich 64Byte E/A-Daten, die im Peripherieadress-Bereich der CPU 24x DP eingeblendet werden.

**Projektierung angekoppelte E/A-Peripherie**

Die System 200V Peripherie-Module, die direkt über den Rückwandbus mit der CPU 24x DP verbunden sind, werden automatisch in den CPU-Adressbereich eingeblendet. Die Adresszuweisung können Sie jederzeit durch Programmierung des DB 1 ändern.

Siehe hierzu auch "Default-Adressierung mit DB 1 ändern"

In Systemen mit mehr als zwei Stationen werden alle Teilnehmer parallel verdrahtet. Hierzu ist das Buskabel unterbrechungsfrei durchzuschleifen.

**Hinweis!**

An den Leitungsenden muss das Buskabel immer mit dem Wellenwiderstand abgeschlossen werden um Reflexionen und damit Übertragungsprobleme zu vermeiden!

**Verbindung mit Profibus**

Profibus verwendet als Übertragungsmedium eine geschirmte, verdrillte Zweidrahtleitung auf Basis der RS485-Schnittstelle.

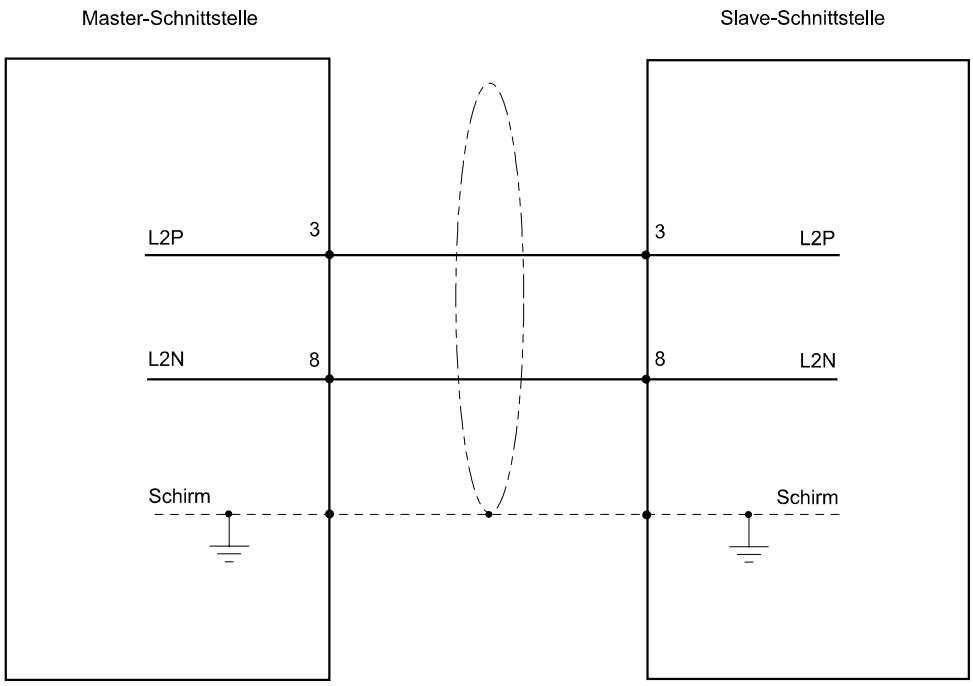
Die RS485-Schnittstelle arbeitet mit Spannungsdifferenzen. Sie ist daher unempfindlicher gegenüber Störeinflüssen als eine Spannungs- oder Stromschnittstelle. Sie können das Netz sowohl als Linien-, als auch als Baumstruktur konfigurieren.

Auf Ihrer VIPA CPU 24x DP befindet sich eine 9-polige Buchse. Über diese Buchse koppeln Sie den Profibus-Koppler als Slave direkt in Ihr Profibus-Netz ein.

Pro Segment sind maximal 32 Teilnehmer zulässig. Die einzelnen Segmente werden über Repeater verbunden. Die max. Segmentlänge ist von der Übertragungsrate abhängig.

Bei Profibus-DP wird die Übertragungsrate aus dem Bereich zwischen 9.6 kBaud bis 12 MBaud eingestellt, Slaves passen sich automatisch an. Alle Teilnehmer im Netz kommunizieren mit der gleichen Baudrate.

Die Busstruktur erlaubt das rückwirkungsfreie Ein- und Auskoppeln von Stationen oder die schrittweise Inbetriebnahme des Systems. Spätere Erweiterungen haben keinen Einfluss auf Stationen, die bereits in Betrieb sind. Es wird automatisch erkannt, ob ein Teilnehmer ausgefallen oder neu am Netz ist.

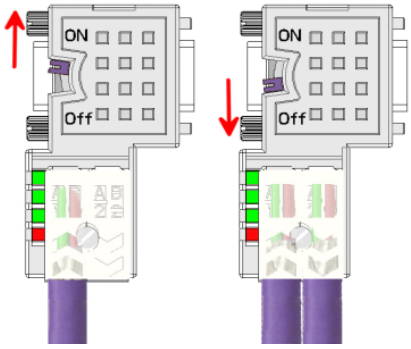



**Achtung!**

An den Leitungsenden muss das Buskabel immer mit dem Wellenwiderstand abgeschlossen werden um Reflexionen und damit Übertragungsprobleme zu vermeiden!

**Leitungs-abschluss**

Auf dem Busanschlussstecker befindet sich unter anderem ein Schalter, mit dem Sie einen Abschlusswiderstand zuschalten können.




**Achtung!**

Der Abschlusswiderstand wird nur wirksam, wenn der Stecker an einem Slave gesteckt ist und der Slave mit Spannung versorgt wird.

**Hinweis!**

Eine ausführliche Beschreibung zum Anschluss und zum Einsatz der Abschlusswiderstände liegt dem Stecker bei.

**Spannungs-versorgung**

Die CPU 24x DP besitzt ein eingebautes Netzteil. Das Netzteil ist mit 24V Gleichspannung zu versorgen. Über die Versorgungsspannung werden neben der CPU und dem Buskoppler auch die angeschlossenen Module über den Rückwandbus versorgt. Bitte beachten Sie, dass das integrierte Netzteil den Rückwandbus mit maximal 3A versorgen kann.

Profibus und Rückwandbus sind galvanisch voneinander getrennt.

**Initialisierungs-phase**

Nach dem Einschalten durchläuft der Profibus-Koppler einen Selbsttest. Hierbei überprüft er seine internen Funktionen, die Kommunikation über den Rückwandbus und die Kommunikation zum Profibus.

Bei erfolgreichem Test werden die DB1-Parameter aus der CPU gelesen und die Profibus-Slave-Parameter geprüft.

Nach fehlerfreiem Hochlauf geht der Buskoppler in den Zustand "READY" über.

Bei Kommunikationsstörungen am Rückwandbus geht der Profibus-Koppler zunächst in STOP und läuft nach ca. 2 Sekunden erneut hoch. Sobald der Test positiv abgeschlossen ist, blinkt die RD-LED.

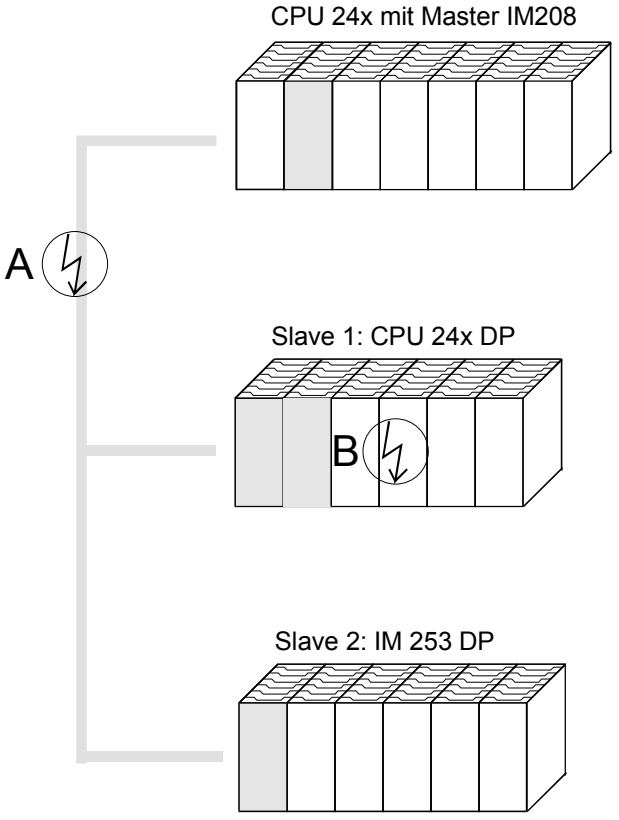
Bei beginnender Kommunikation leuchtet die DE-LED.

**Achtung!**

Bei fehlerhaften DB1-Parametern geht die CPU in Stop!

# Einsatz der Diagnose-LEDs

Das folgende Beispiel zeigt die Reaktion der LEDs bei unterschiedlichen Netzwerkunterbrechungen.




**Unterbrechung Position A**

Der Profibus ist unterbrochen.

**Unterbrechung Position B**

Die Kommunikation über den Rückwandbus ist unterbrochen.

| LED CPU 24x DP | Unterbrechung Position | |
|---|---|---|
| LED | A | B |
| RD | blinkt | aus |
| ER | aus | an |
| DE | aus | aus |
| ST | aus | an |
| QV | aus | an |
| BASP | aus | an |

| LED IM 253 DP | Unterbrechung Position | |
|---|---|---|
| LED | A | B |
| RD | blinkt | an |
| ER | aus | aus |
| DE | aus | an |

Die LEDs sind in der nachfolgenden Tabelle aufgeführt

| Bezeichnung | Farbe | Bedeutung |
|---|---|---|
| PW | Grün | Signalisiert eine anliegende Betriebsspannung |
| ER | Rot | Fehler am Rückwandbus (Unterspannung):<br>blinkt abwechselnd mit RD wenn DB1-Konfiguration nicht mit der Master-Konfiguration übereinstimmt. |
| RD | Grün | Mit Hilfe von RD (READY) wird ein positiver Selbsttest angezeigt. |
| DE | Grün | DE (Data exchange) zeigt an, dass eine Kommunikation mit dem Profibus stattfindet. |
| ST | Rot | CPU befindet sich in STOP. |
| QV | Rot | Quittungsverzugszeit wurde überschritten. |
| BA | Rot | Befehlsausgabesperre (BASP) ist aktiviert, d.h. Ausgänge sind gesperrt. |

# Beispiel

**Aufgabenstellung**

In diesem Beispiel soll eine Kommunikation zwischen zwei CPUs über den Profibus gezeigt werden.

Hierbei sollen Zählerstände über den Profibus ausgetauscht und diese auf dem Ausgabe-Modul des jeweiligen Partners dargestellt werden.

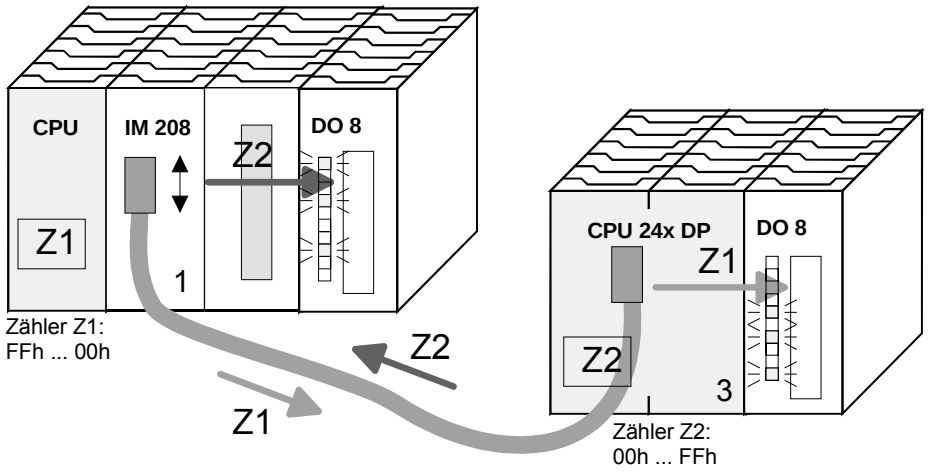
**Aufgabenstellung im Detail**

Die CPU des Masters soll von FFh ... 00h zählen und den Zählerstand zyklisch in den Ausgabebereich des Profibus-Masters übertragen. Der Master hat diesen Wert an den Slave der CPU 24x DP zu schicken.

Der empfangene Wert soll in der CPU im Eingangs-Peripheriebereich abgelegt und über den Rückwandbus auf dem Ausgabe-Modul ausgegeben werden.

Umgekehrt soll die CPU 24x DP von 00h bis FFh zählen. Auch dieser Zählerstand ist im Ausgabe-Bereich des CPU-Slaves abzulegen und über den Profibus in den Master zu transferieren.

Dieser Wert ist auf dem Ausgabe-Modul des Masters auszugeben.




**Projektierdaten**

**CPU 24x mit IM 208**

Zählerstand: MB 0 (FFh ... 00h)

| | | |
|---|---|---|
| Profibus-Adresse: | 1 | |
| Eingangsbereich: | PY 10 | Länge: 2 Byte |
| Ausgangsbereich: | PY 20 | Länge: 2 Byte |

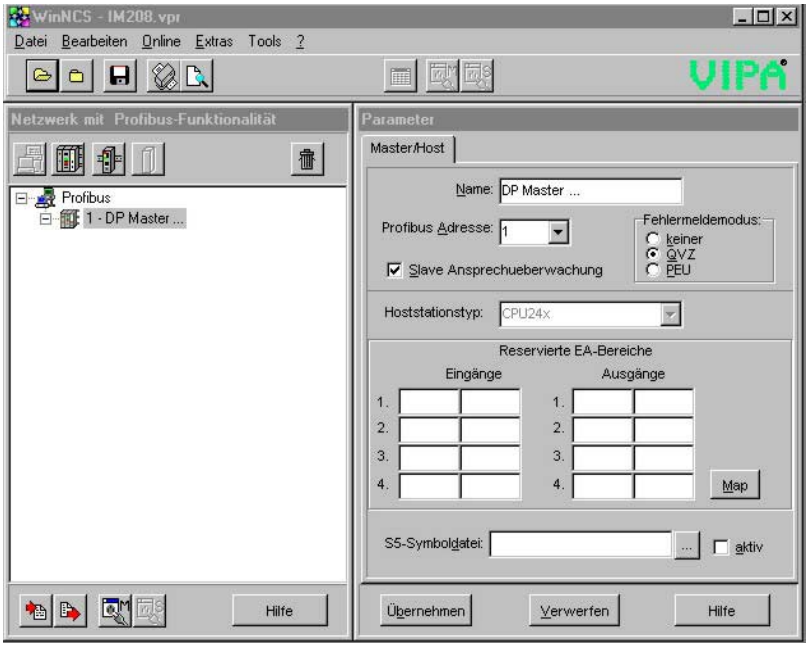
**CPU 24x DP**

Zählerstand: MB 0 (00h ... FFh)

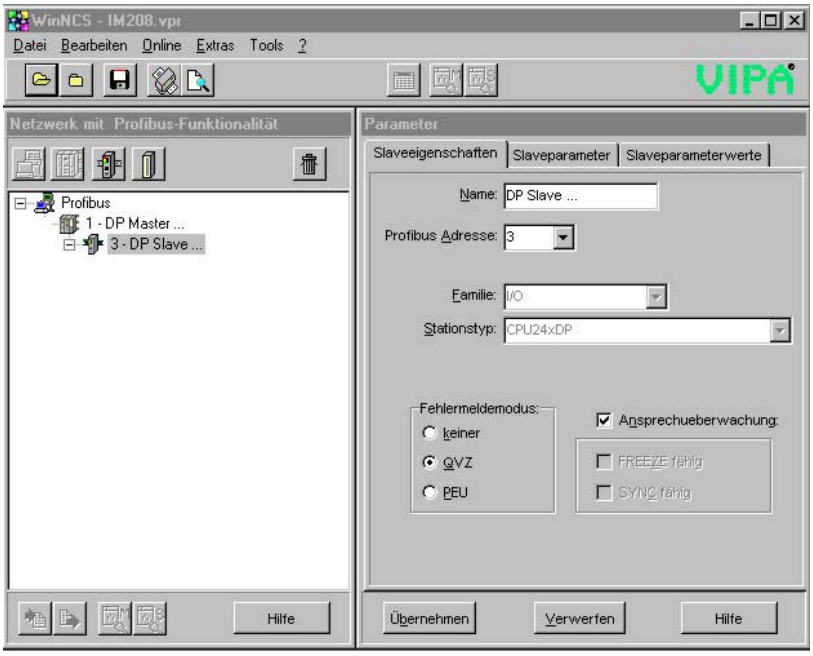
| | | |
|---|---|---|
| Profibus-Adresse: | 3 | |
| Eingangsbereich: | PY 30 | Länge: 2 Byte |
| Ausgangsbereich: | PY 40 | Länge: 2 Byte |
| Parameterdaten: | PY 50 | Länge: 24 Byte (fest) |
| Diagnosedaten: | PY 60 | Länge: 6 Byte (fest) |
| Statusdaten: | PY 100 | Länge: 2 Byte (fest) |

**Projektierung CPU 24x mit IM 208**

- Starten Sie WinNCS
- Stellen Sie unter **Extras** die Funktionalität "Profibus" ein
- Fügen Sie die Funktionsgruppe "Profibus" ein
- Projektieren Sie hier Ihre Profibus-Master Baugruppe IM 208 indem Sie im Netzwerkfenster über "Profibus Host/Master" einen Master einfügen.
- Stellen Sie im Dialogfenster "Parameter" wie unten gezeigt folgende Werte ein:




- Binden Sie Ihre CPU 24x DP ein, indem Sie im Netzwerkfenster einen "Profibus Slave" einfügen und unter "Parameter" im Register Slaveeigenschaften folgende Werte einstellen:



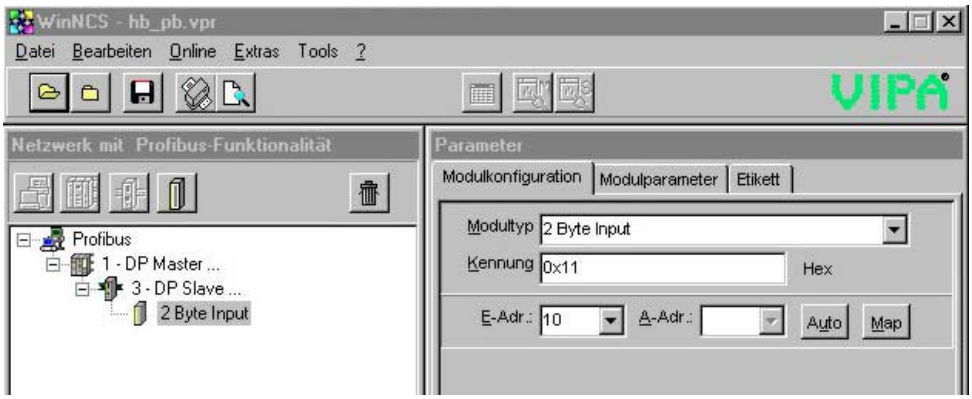
**Hinweis!**

Bitte beachten Sie, dass die am Adresseinsteller gewählte Profibus-Adresse mit Ihrer projektierten Adresse übereinstimmt!

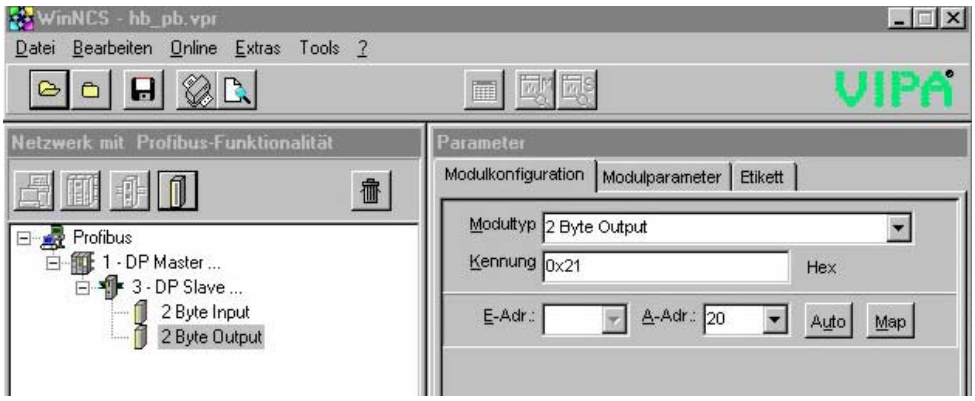
- **Ein-/Ausgabe-Bereiche Profibus**

  Die Module, die über den Rückwandbus direkt mit der CPU 24x DP verbunden sind, können unter Profibus als Modul nicht projektiert werden. Bei der CPU 24x DP werden die Daten über Ein- bzw. Ausgabebereiche transferiert. Die CPU kann Daten in diese Bereiche schreiben und hieraus Daten entnehmen.

  In diesem Beispiel haben Ein- und Ausgabe-Bereich des Profibus-Slaves je eine Größe von 2Byte. Fügen Sie hierzu die Modultypen "2 Byte Input" und "2 Byte Output" ein. Das Eingabe-Byte soll in PY 10 abgelegt werden:




Das Ausgabe-Byte des Profibus-Masters legen Sie auf Adresse PY 20:




Hiermit ist die Projektierung unter WinNCS abgeschlossen.

Es folgen nun die Anwenderprogramme in den beiden CPUs.

**Anwender-programm in CPU 24x**

Das Anwenderprogramm in der CPU 24x hat zwei Aufgaben, die auf zwei OBs verteilt werden:

- Über Kontrollbyte die Kommunikation testen.
  Vom Profibus das Eingangs-Byte laden und den Wert auf dem Ausgabe-Modul ausgeben.

  OB 1 (zyklischer Aufruf)

```
:L    KH 00FF
:T    AB  20           Kontrollbyte für Slave-CPU
:
:L    KH 00FE          Kontrollwert 0xFE laden
:L    EB  10           Wurde Kontrollbyte von der Slave
:><F                   CPU richtig übermittelt?
:BEB                   Nein -> Ende
:
:                      -------------------------------
:                      Datenaustausch via Profibus
:
:L    EB  11           Lade Eingangsbyte 11 (Ausgangsdaten
:                      der CPU24xDP) und
:T    AB   0           transferiere ins Ausgangsbyte 0
:
:BE
```

- Zählerstand aus dem MB 0 lesen, dekrementieren, in MB 0 speichern und über Profibus an CPU 24x DP ausgeben.

  OB 13 (Zeit-OB)

```
:L    MB   0           Zaehler von 0xFF bis 0x00
:L    KH 0001
:-F
:T    MB   0
:
:T    AB  21           Transferiere ins Ausgangsbyte 21
:                      (Eingangsdaten der CPU24xDP)
:BE
```

Auf Seiten der CPU 24x und IM 208 DP ist jetzt alles programmiert. Auch die Profibus-Kommunikation ist nun auf beiden Seiten festgelegt.

Es fehlt lediglich das Anwenderprogramm für die CPU 24x DP.

**Anwender-programm in CPU 24x DP**

Das Anwenderprogramm hat wie schon weiter oben gezeigt zwei Aufgaben, die auch bei dieser CPU auf zwei OBs verteilt werden:

- Vom Profibus-Slave das Eingangs-Byte laden und den Wert auf dem Ausgabe-Modul ausgeben.

OB 1 (zyklischer Aufruf)

```
:L    PW 100           Statusdaten laden und in Merker-
:T    MW 100           wort speichern
:
:UN   M  100.5         Inbetriebnahme durch DP-Master
:BEB                   erfolgt? Nein -> Ende
:
:U    M  101.4         Empfangsdaten gültig?
:BEB                   Nein -> Ende
:L    KH 00FF          Kontrollwert laden und mit
:L    PY  30           Kontrollbyte (1. Eingangsbyte)
:><F                   vergleichen
:BEB                   Empfangene Daten haben keine
:                      gültigen Werte
:
:L    KH 00FE          Kontrollbyte für Master-CPU
:T    PY  40
                       ------------------------------
:                      Datenaustauch via Profibus
:
:L    PY  31           Lade Peripheriebyte 31 (Eingangs
:                      daten vom Profibus-Slave) und
:T    AB   0           transferiere ins Ausgangsbyte 0
:
:BE
```

- Zählerstand aus dem MB 0 lesen, inkrementieren, in MB 0 speichern und über Profibus an CPU 24x ausgeben.

OB 13 (Zeit-OB)

```
:L    MB   0           Zähler von 0x00 bis 0xFF
:L    KH 0001
:+F
:T    MB   0
:
:T    PY  41           Transferiere Zählerwert ins
:                      Peripheriebyte 41 (Ausgangsdaten
:                      des Profibus-Slaves)
:BE
```

- Durch Programmierung des DB 1 mit dem Label DPS bestimmen Sie in der CPU die Adress-Bereiche des Profibus-Slaves.

  Folgende Adress-Bereiche werden hier über den DB 1 eingestellt.

  | | | |
  |---|---|---|
  | Eingangsbereich: | PY 30 | Länge: 2 Byte |
  | Ausgangsbereich: | PY 40 | Länge: 2 Byte |
  | Parameterdaten: | PY 50 | Länge: 24 Byte (fest) |
  | Diagnosedaten: | PY 60 | Länge: 6 Byte (fest) |
  | Statusdaten: | PY 100 | Länge: 2 Byte (fest) |

  DB 1

```
0:     KC ='DB1 TFB: OB13 10  ; SDP:';
12:    KC =' WD 500 ; DPS: 30 2 40 2';
24:    KC =' 50 60 100 ; END ;';
```

**DB1-Parametrierung unter WinNCS**

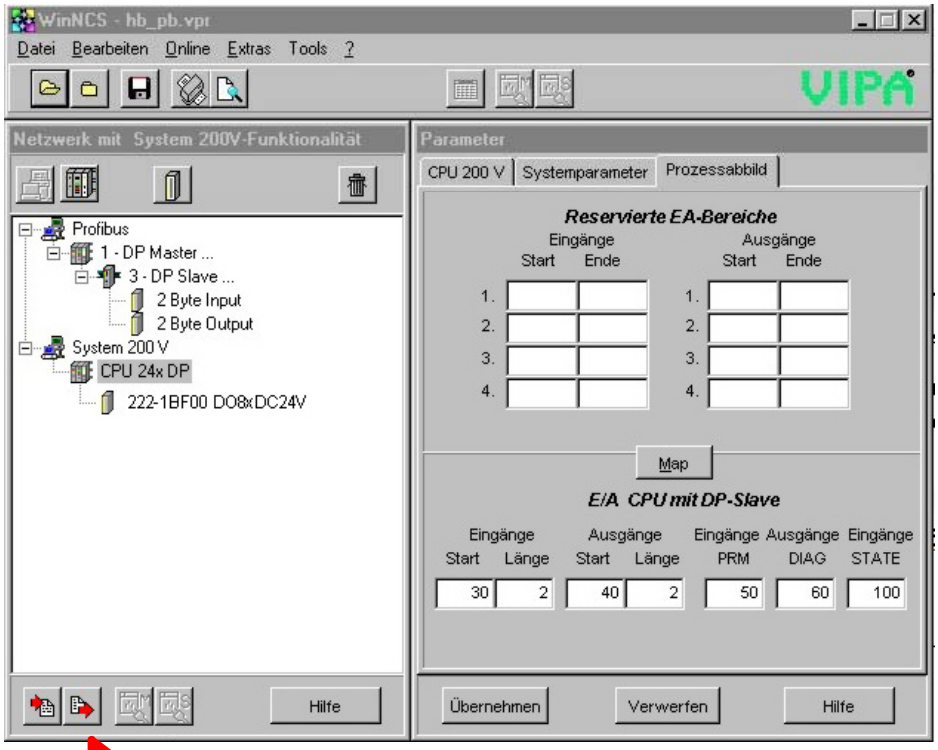
WinNCS bietet Ihnen die Möglichkeit für die DB1-Parametrierung

Starten Sie hierzu WinNCS in der Funktionalität *System 200V*.

Projektieren Sie eine CPU 24x DP und geben Sie im Register *Prozessabbild* die Adressbereiche für den Profibus-Slave an.

Fügen Sie unterhalb der CPU das Ausgabe-Modul an, das im Beispiel direkt neben der CPU 24x DP steckt und weisen Sie diesem eine Adresse zu. Bestätigen Sie alle Eingaben mit [Übernehmen]

WinNCS bildet einen DB 1 nach, den Sie unter *Systemparameter* darstellen und als s5d-Datei exportieren können.




Adressbereiche Profibus-Slave

DB1-Export in s5d-Datei

# Teil 7 Betriebszustände

**Überblick**

In diesem Kapitel werden die Betriebszustände STOP, ANLAUF und RUN beschrieben.

Es informiert Sie ausführlich über die verschiedenen Anlaufarten und den damit verbundenen Organisationsbausteinen, in denen Sie Ihr spezifisches Programm für verschiedene Anlauffälle programmieren können.

Sie erfahren ferner, wodurch sich die Programmbearbeitungen "Zyklische Bearbeitung", "Zeitgesteuerte Bearbeitung" und "Alarmgesteuerte Bearbeitung" auszeichnen und welche Bausteine für Ihr Anwenderprogramm hierbei zur Verfügung stehen.

**Inhalt**

**Thema** **Seite**

**Teil 7 Betriebszustände ..... 7-1**

Einführung und Übersicht ..... 7-2

Programmbearbeitungsebenen ..... 7-3

Betriebszustand STOP ..... 7-6

Betriebszustand ANLAUF ..... 7-8

Betriebszustand RUN ..... 7-11

Programmablauf ..... 7-12

# Einführung und Übersicht

**Betriebszustände**

Die CPU 24x kennt drei Betriebszustände:

- Betriebszustand STOP
- Betriebszustand ANLAUF
- Betriebszustand RUN

In den Betriebszuständen ANLAUF und RUN können bestimmte Ereignisse auftreten, auf die das Systemprogramm reagieren muss. In vielen Fällen wird dabei ein für das Ereignis vorgesehener Organisationsbaustein (einer von OB 1 bis OB 31) als Anwenderschnittstelle aufgerufen.

Die Betriebszustände werden über LEDs auf der Frontplatte der CPU angezeigt.

Einige der Betriebszustände müssen Sie per Bedienung aktivieren. Dazu können Sie u.a. die Bedienungselemente auf der Frontplatte der CPU benutzen.

**LED-Anzeige der Betriebszustände und Fehler**

**Betriebszustände**

Auf der CPU-Frontplatte signalisieren Ihnen verschiedene LEDs den aktuellen Betriebszustand der CPU. Nachfolgende Tabelle zeigt den Zusammenhang zwischen den Anzeigen der STOP- und RUN-LED und dem zugehörigen Betriebszustand.

Diese Anzeigen werden ergänzt durch weitere LED-Anzeigen (BASP, QVZ, ZYK).

Die Bedeutung der LEDs "RUN" und "STOP" finden Sie in der Tabelle:

*Betriebszustände*

| RUN | STOP | Betriebszustand |
|---|---|---|
| an | aus | Die CPU ist im Betriebszustand RUN |
| aus | an | Die CPU ist im Betriebszustand STOP |
| an | an | Die CPU ist im Betriebszustand ANLAUF oder<br>CPU ist in ANLAUF/RUN, die Bearbeitungskontrolle ist aktiviert und ein Haltepunkt ist erreicht, oder<br>CPU ist in ANLAUF/RUN, die Bearbeitungskontrolle ist aktiviert und ein Haltepunkt wurde vor dem Erreichen wieder ausgetragen (Wartezustand) |

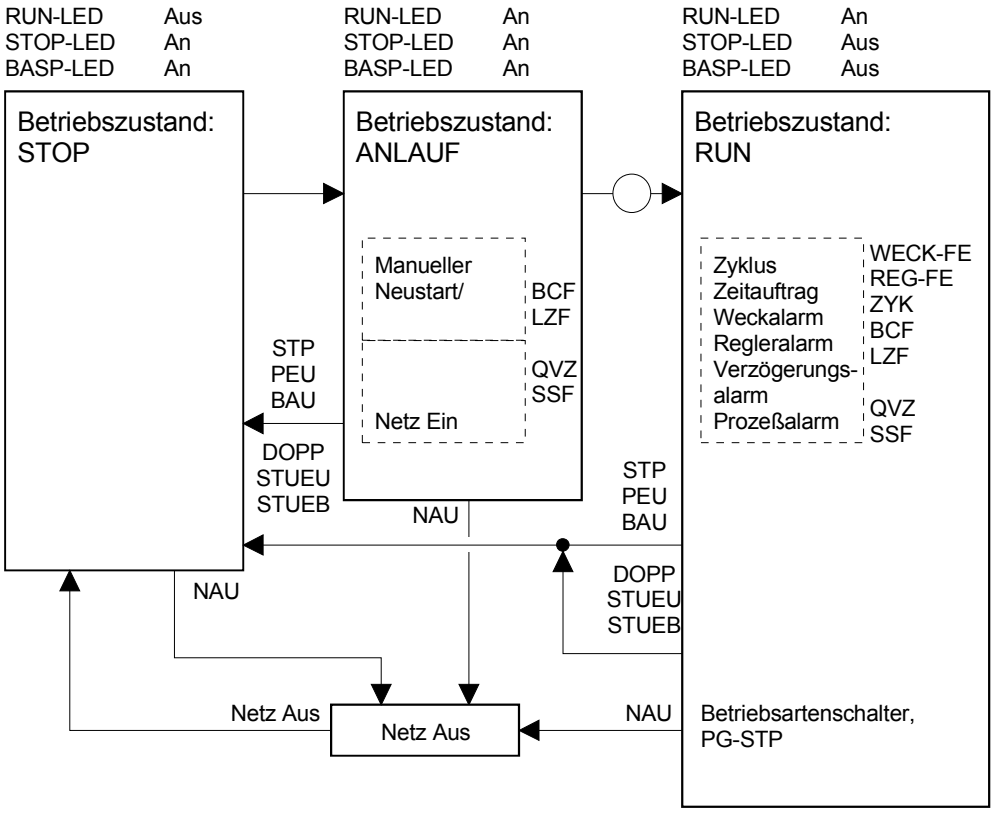
**Systemfehler**

Während des Betriebs erfolgt eine ständige Überwachung von besonders wichtigen und internen Funktionen. Eine Abweichung dieser internen Funktionen vom Normalbetrieb wird erkannt und als Systemfehler behandelt.

Die genauere Spezifikation der Ursache für einen Systemfehler können Sie dem Systemdatenwort entnehmen.

# Programmbearbeitungsebenen

Nachfolgend finden Sie eine Übersicht der Betriebszustände und Bearbeitungsebenen der CPU 24x.




Erläuterung zu den Abkürzungen:

| | |
|---|---|
| STP | = Stop-Anweisung |
| PEU | = Peripherie unklar |
| BAU | = Batterie unklar, Batterie leer oder fehlt |
| DOPP | = Doppelfehler ( Eine noch aktive Fehlerprogrammbearbeitungsebene wird ein zweites Mal aktiviert - Neustart erforderlich) |
| STUEU | = USTACK - Überlauf |
| STUEB | = BSTACK - Überlauf |
| NAU = | = Netzausfall |
| PG-STP | = Stop Anwahl am PG |

**Programm-bearbeitungs-ebenen im ANLAUF**

Manueller Neustart
Netz Ein
BCF (Befehlscodefehler)

→ Anlaufebene

LZF (Laufzeitfehler)
QVZ (Quittungsverzug)
SSF (Schnittstellenfehler)

→ Fehlerebene

**Programm-bearbeitungs-ebenen im RUN**

Zyklus
Zeitauftrag
4 Weckalarme einstellbar (zeitgesteuert)

→ Grundebene

Regleralarm (zeitgesteuert)
Verzögerungsalarm (zeitgesteuert)
Prozessalarm (alarmgesteuert)

→ Grundebene

WECK-FE (Weckfehler)
REG-FE (Reglerfehler)
ZYK (Zyklusfehler)
BCF (Befehlscodefehler)
LZF (Laufzeitfehler)
QVZ (Quittungsverzugfehler)
SSF (Schnittstellenfehler)

→ Fehlerebene

**Merkmale einer Programmbearbeitungsebene**

Eine Programmbearbeitungsebene ist durch bestimmte Merkmale charakterisiert, die hier kurz zusammengefasst sind.

**Einschachtelung von anderen Ebenen**

Bei einem Ereignis, das eine höherpriore Bearbeitung erfordert, wird die aktuelle Ebene vom Systemprogramm unterbrochen und die höherpriore Ebene eingeschachtelt.

**Spezifisches Systemprogramm**

Jede Programmbearbeitungsebene hat ihr eigenes Systemprogramm.

**USTACK**

Nach dem Aufruf eines OBs werden die SPS-Anweisungen ausgeführt. Hierbei wird der aktuelle Registersatz in den USTACK gerettet und ein neuer Registersatz angelegt.

Ein Registersatz besteht aus: AKKU 1...2, Bausteinstack-Pointer, Baustein-Adressregister, Datenbaustein-Anfangsadresse/-Länge, STEP-Adresszähler und Basisadressregister.

**Priorität**

Die Programmbearbeitungsebenen besitzen eine feste Priorität. Abhängig von der Priorität können sich diese gegenseitig unterbrechen bzw. ineinander verschachtelt werden.

Grundebenen können sich nur an Bausteingrenzen einschachteln, Anlauf und Fehlerebenen werden, sobald das Ereignis aufgetreten ist, an der nächsten Befehlsgrenze eingeschachtelt.

Bei Fehlern hat der zuletzt aufgetretene immer höchste Priorität.

**Reaktion bei einem Doppelfehler**

Eine einmal aktivierte Fehlerebene, die noch nicht vollständig abgearbeitet ist, kann nicht nochmals aktiviert werden. In diesem Fall geht die CPU in unmittelbaren STOP. Eine Ausnahme bilden hier die Weckfehler.

Im USTACK mit Tiefe "01" ist die Kennung DOPP und die aufgerufene Fehlerebene angekreuzt.

# Betriebszustand STOP

**Kennzeichen**

Der Betriebszustand STOP ist durch folgende Merkmale charakterisiert:

**Anwenderprogramm**

Das Anwenderprogramm wird nicht bearbeitet.

**Erhalt von Daten**

Hat zuvor eine Programmbearbeitung stattgefunden, dann bleiben die Werte von Zählern, Zeiten, Merkern und des Prozessabbilds beim Übergang in den STOP-Zustand erhalten.

**BASP-Signal**

Die Befehlsausgabe ist gesperrt d.h. alle digitalen Ausgaben sind gesperrt.

**USTACK**

Hat zuvor eine Programmbearbeitung stattgefunden, so finden Sie im USTACK einen Informationsblock mit der Unterbrechungsursache.

**LED-Anzeigen**

| | |
|---|---|
| RUN-LED | Aus |
| BASP-LED | An |

**STOP-LED An**

Betriebszustand STOP wurde ausgelöst durch:

- RUN/STOP-Schalter auf STOP
- PG-Funktion AG-STOP
- PG-Funktion "Bearbeitungskontrolle Ende"
- Gerätefehler (BAU, PEU)
- Urlöschen

## Urlöschen

Das Urlöschen bringt die CPU wieder in den Auslieferungszustand. Programm, Daten, Register und Zeiger werden gelöscht und falls ein EPROM bzw. eine Flash-Card gesteckt ist, werden nach dem Löschen die Daten in die CPU übertragen. Bevor das Urlöschen ausgeführt werden kann, muss es angefordert werden. Bei Betrieb mit EPROM ist ein Neustart mit Rücksetzen möglich.

### Urlöschen

Urlöschen kann vom System oder vom Anwender angefordert werden.

**Anforderung vom Systemprogramm**

Nach jedem Einschalten der Netzspannung und nach dem Urlöschen wird eine Initialisierungsroutine durchlaufen. Wird hierbei ein Fehler bemerkt, so geht die CPU in den STOP-Zustand über.

Die Störursache ist dann in jedem Fall zu beseitigen. Sollte dies nicht möglich sein, dann wenden Sie sich an die VIPA-Hotline.

**Anforderung vom Anwender**

Durch folgende Bedienschritte fordern Sie das Urlöschen an:

Bringen Sie den RUN/STOP-Schalter in Stellung STOP (STOP-LED zeigt Dauerlicht).

- Halten Sie den Betriebsartentaster in Stellung "OR" fest, und stellen Sie gleichzeitig den Betriebsartenschalter von "ST" nach "RN".
- Diesen Vorgang mehrmals wiederholen. Dabei erlischt die rote LED "ST" kurz.

  → Die CPU ist urgelöscht und die rote LED "ST" zeigt Dauerlicht.

Sie können die CPU auch vom PG aus löschen. Verwenden Sie hierzu die PG-Funktion URLÖSCHEN. Die manuelle Löschanforderung kann entfallen. Die Stellung des Betriebsartenschalters ist hierbei irrelevant.

**Hinweis!**

Nach dem Urlöschen ist nur die Anlaufart NEUSTART zulässig.

# Betriebszustand ANLAUF

## Kennzeichen

Folgende Eigenschaften sind kennzeichnend für die Betriebsart Anlauf:

**Organisationsbausteine (OBs)**

Die Länge der Organisationsbausteine ist nicht beschränkt. Auch wird der Ablauf zeitlich nicht überwacht. In den Anlauf OBs können weitere Bausteine aufgerufen werden.

Folgende Organisationsbausteine werden aufgerufen:

| | |
|---|---|
| Neustart | OB 21 |
| Automatischer Wiederanlauf | OB 22 |

**Datenbehandlung**

Die Werte von Zählern, Zeiten, Merkern und Prozessabbildern werden in den einzelnen Anlaufarten unterschiedlich behandelt.

**BASP-Signal**

Beim Anlauf sind alle digitalen Ausgaben gesperrt, d.h. die Befehlsausgabesperre ist aktiv.

**LEDs auf der Frontplatte**

| | |
|---|---|
| RUN-LED | an |
| STOP-LED | an |
| BASP-LED | an |

**Anlaufarten**

In der Betriebsart Anlauf gibt es folgende zwei grundsätzlichen Arten des Anlaufs, die wiederum untergliedert sind:

- Neustart
- Wiederanlauf

## Neustart

Alle Merker, Zeiten, Zähler und das Prozessabbild werden gelöscht, falls "nichtremanent" eingestellt wurde. Die Bearbeitung des Anwenderprogramms beginnt von vorn.

Ein Neustart ist immer zulässig, sofern vom Systemprogramm nicht Urlöschen angefordert wird.

**Neustart mit Rücksetzen von Hand (manueller Neustart)**

Bringen Sie den RUN/STOP-Schalter in Stellung RUN. Die CPU führt daraufhin einen manuellen Neustart durch und zeigt mit der RUN-LED den RUN-Zustand an.

## Wiederanlauf

Die Zustände der Merker, Zeiten und Zähler bleiben während der Stillstandszeit erhalten.

Wiederanlauf ist nicht zulässig, wenn Urlöschen angefordert ist oder eins der folgenden Ereignisse eingetreten ist:

- Doppelaufruf einer Programmbearbeitungsebene,
- Urlöschen (Steuerbits: URGELOE),
- Anlaufabbruch (Steuerbits: ANL-ABB),
- STOP nach PG-Funktion BEARBEITUNGSKONTROLLE ENDE,
- Komprimieren im STOP,
- Stacküberlauf,
- Änderung des Anwenderprogramms im STOP.

**Wiederanlauf automatisch**

Nach Wiederkehr der Versorgungsspannung geht das Gerät selbständig wieder in den Betriebszustand RUN über.

Voraussetzung für den automatischen Wiederanlauf:

- RUN/STOP-Schalter befindet sich noch immer in Stellung RUN,
- CPU befindet sich vor dem Spannungsausfall im RUN,
- Bei der Initialisierung sind keine Fehler aufgetreten.

## Anwender-schnittstellen für den Anlauf

Als Anwenderschnittstellen für die verschiedenen Anlaufarten dienen die Organisationsbausteine OB 21 und OB 22. In diesen Bausteinen können Sie Ihr Programm für die verschiedenen Anlaufarten hinterlegen. In den Anlauf-OBs können Sie z.B. Merker setzen, Zeiten starten, Datenverkehr CPU mit Peripherie vorbereiten und die Synchronisation von mehreren CPs durchführen.

Die OBs haben folgende Funktionen:

**OB 21 - Neustart**

Sobald die CPU einen manuellen Neustart durchführt, wird vom Systemprogramm einmal der OB 21 aufgerufen. Legen Sie in OB 21 ein Programm ab, das vorbereitende Schritte für einen Neustart der zyklischen Bearbeitung durchführt.

Nach der Bearbeitung des OB 21 beginnt die zyklische Programmbearbeitung durch Aufruf von OB 1.

**OB 22 - Automatischer Wiederanlauf**

Bei automatischem Wiederanlauf ruft das Systemprogramm den OB 22 auf. Legen Sie in OB 22 ein Programm ab, das den Wiederanlauf eines zuvor im RUN befindlichen Programms regelt. Nach dem OB 22 wird die zyklische Programmbearbeitung aufgerufen.

Hierbei gilt wie beim manuellen Wiederanlauf:

- Das BASP-Signal bleibt während der Bearbeitung des Restzyklus aktiv und wird erst deaktiviert, wenn ein neuer Zyklus beginnt.
- Das Prozessabbild bleibt am Ende des Restzyklus zurückgesetzt.

## Unterbrechungen im Anlauf

Ein Anlaufprogramm kann unterbrochen werden durch:

- Netzspannungsausfall im Zentralgerät oder im Erweiterungsgerät,
- STOP (Schalter, Befehl),
- Programm- und Gerätefehler.

# Betriebszustand RUN

**Merkmale**

Wenn die CPU einen Anlauf fertig bearbeitet hat, geht Sie in den Betriebszustand RUN über. Folgende Merkmale charakterisieren diesen Zustand:

**Bearbeitung des Anwenderprogramms**

Das Anwenderprogramm im OB 1 wird zyklisch bearbeitet, wobei zusätzlich alarmgesteuert weitere Programmteile eingeschachtelt werden können.

**Zeiten, Zähler, Prozessabbild**

Alle im Programm gestarteten Zeiten und Zähler laufen und das Prozessabbild wird zyklisch aktualisiert.

**BASP-Signal**

Das BASP-Signal (Befehlsausgabesperre) wird deaktiviert d.h. alle digitalen Ausgänge sind freigegeben.

**LEDs auf der Frontplatte**

| | |
|---|---|
| RUN-LED | an |
| STOP-LED | aus |
| BASP-LED | aus |

**Programmbearbeitungsebenen**

Die Programmbearbeitungsebenen werden durch unterschiedliche Ereignisse ausgelöst. Für jede Bearbeitungsebene stehen Ihnen als Anwenderschnittstelle eigene OBs und FBs zur Verfügung. In der CPU 24x können alle Grundprogrammbearbeitungsebenen gleichzeitig programmiert sein.

Abhängig von der Priorität erfolgt der Aufruf der entsprechenden Ebene aus dem Systemprogramm heraus. Im Betriebszustand RUN bietet Ihnen die CPU folgende Grundprogrammbearbeitungsebenen:

- Zyklus
  Das Anwenderprogramm wird zyklisch bearbeitet.
- Zeitauftrag
  In einem definierbaren Zeitintervall wird Ihr Anwenderprogramm bearbeitet. Sie können aber auch einen festen Zeitpunkt vorgeben, zu dem das Anwenderprogramm einmalig bearbeitet werden soll.
- 1 Weckalarm

  Das Anwenderprogramm wird in festen, vom System angebotenen Zeitintervallen bearbeitet.
- Prozessalarm
  Das Anwenderprogramm wird alarmgesteuert bearbeitet.

# Programmablauf

## Zyklische Programm-bearbeitung

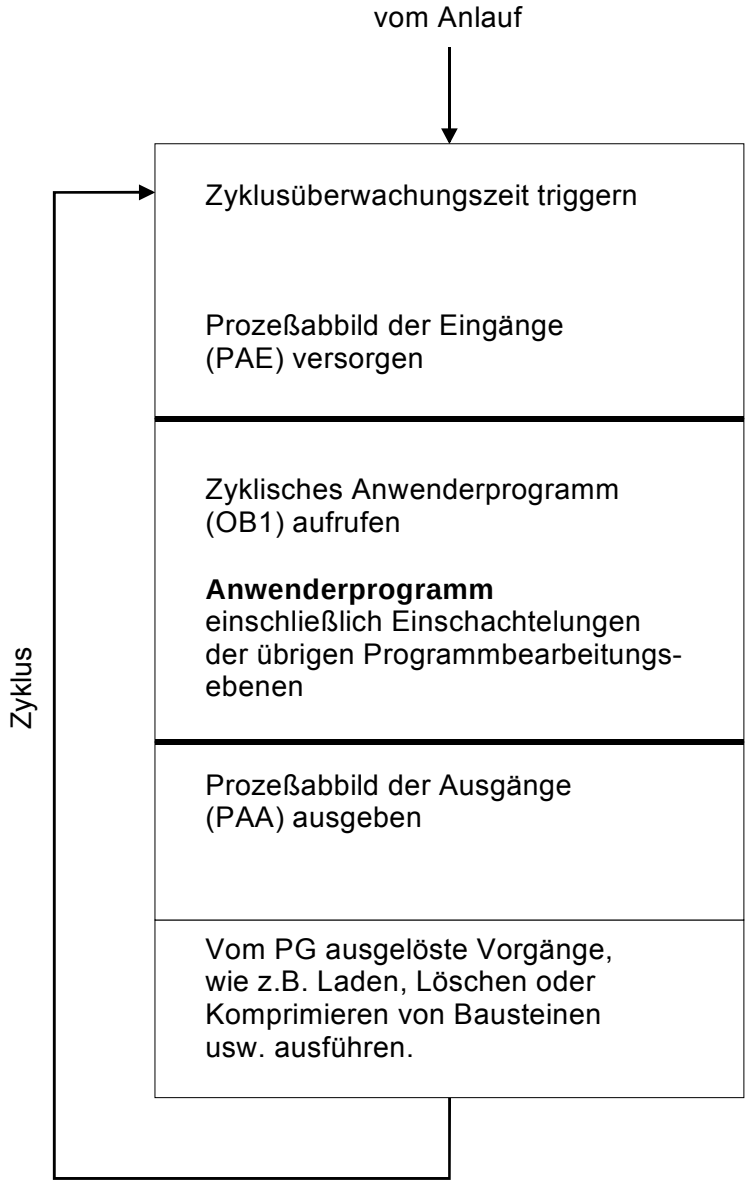
Im allgemeinen wird bei der SPS die zyklische Programmbearbeitung eingesetzt (Programmbearbeitungsebene ZYKLUS). Es handelt sich hierbei um einen freien Zyklus. Nach Erreichen des Zyklusendes beginnt sofort die nächste zyklische Bearbeitung.

**Auslösung**

Hat die CPU das Anlaufprogramm fehlerfrei abgearbeitet, so beginnt sie mit der zyklischen Programmbearbeitung.

**Prinzip**

Das Prinzip der zyklischen Programmbearbeitung soll folgende Abbildung verdeutlichen:


vom Anlauf
Zyklusüberwachungszeit triggern
Prozeßabbild der Eingänge (PAE) versorgen
Zyklisches Anwenderprogramm (OB1) aufrufen
Anwenderprogramm einschließlich Einschachtelungen der übrigen Programmbearbeitungs-ebenen
Prozeßabbild der Ausgänge (PAA) ausgeben
Vom PG ausgelöste Vorgänge, wie z.B. Laden, Löschen oder Komprimieren von Bausteinen usw. ausführen.
Zyklus

**Anwenderschnittstelle OB1**

Bei der zyklischen Programmbearbeitung wird als Anwenderschnittstelle regelmäßig der OB 1 aufgerufen.

Das Anwenderprogramm im OB 1 wird von Anfang an über die darin von Ihnen programmierten Bausteinaufrufe hinweg durchgehend bearbeitet.

**Unterbrechungsstellen**

Die zyklische Programmbearbeitung kann an den Bausteingrenzen unterbrochen werden durch:

- prozessalarmgesteuerte Programmbearbeitung,
- Reglerbearbeitung,
- zeitgesteuerte Programmbearbeitung.

Die zyklische Programmbearbeitung kann an den Befehlsgrenzen unterbrochen bzw. ganz abgebrochen werden durch:

- Auftreten eines Geräte- oder Programmfehlers,
- Bedienung (PG-Funktion, STOP-Schalter),
- STOP-Befehl.

**AKKUs als Datenspeicher**

Sie können bei der CPU 24x die Rechenregister AKKU 1 bis 2 über Zyklusgrenzen hinweg (Ende Programmzyklus bis zum Beginn des nächsten) als Datenspeicher verwenden.

### Alarmgesteuerte Programmbearbeitung

Zur alarmgesteuerten Programmbearbeitung seht in der CPU 24x-Serie der OB 2 zur Verfügung. Ein Modul im System 200V kann über eine Busleitung einen Interrupt in der CPU auslösen. Dieser Interrupt hat zur Folge, dass der OB 2 an der nächsten Befehlsgrenze aufgerufen wird. Der Alarmbaustein kann eine laufende zeitgesteuerte Programmbearbeitung unterbrechen. Ein zeitdauergesteuertes Programm, also der OB 6, kann nicht unterbrochen werden. Weitere Alarme, die während der Bearbeitung auftreten gehen verloren. Um den aufgetretenen Alarm zuordnen zu können darf nur eine alarmfähige Baugruppe im System vorhanden sein.

### Zeitdauergesteuerte Programmbearbeitung

Die zeitdauergesteuerte Programmbearbeitung wird vom OB 6 übernommen. Der Aufruf der zeitdauergesteuerten Programmbearbeitung wird mit dem Systemdatum BS 101 koordiniert.

Wird ein Wert in das Systemdatum transferiert, so startet damit die Zeit bis zum Aufruf des OB 6. Die Zahl in der BS-Zelle gibt die Zeit in ms an. Beim Aufruf des OB 6 wird die BS-Zelle gelöscht. Die Zeit ist nachtriggerbar, d.h. wird erneut ein Wert in die BS-Zelle transferiert, so wird die Zeit neu mit dem entsprechenden Wert gestartet. Der Aufruf bzw. der Zeitablauf kann gestoppt werden, indem man den Wert Null in die BS-Zelle schreibt.

Die zeitdauergesteuerte Programmbearbeitung kann weder durch eine zeitgesteuerte noch durch eine alarmgesteuerte Programmbearbeitung unterbrochen werden.

## Zeitgesteuerte Programmbearbeitung

Bei einer zeitgesteuerten Programmbearbeitung wird die CPU von einem internen Takt oder von einem Zeitsignal veranlasst, die aktuelle Programmbearbeitung zu unterbrechen und ein spezifisches Programm zu bearbeiten. Nach der Bearbeitung kehrt die CPU wieder zurück an die Unterbrechungsstelle. Auf diese Weise können Sie zeitgesteuert Programmabschnitte einschieben.

*Weckalarme*

Auslösung in einstellbaren Zeitrastern 10ms ... 5s durch Weckalarme. Den Weckalarmen sind OB's zugeordnet (OB 10 ... OB13). Hierbei handelt es sich um feste Zyklen, d.h. die Zeitdauer zwischen zwei Programmstarts ist fest.

**Achtung!**

Bei der zeit- oder alarmgesteuerten Bearbeitung besteht die Gefahr, dass Merker überschrieben werden. Sie können die Merker zu Beginn einer alarm- bzw. zeitgesteuerten Bearbeitung retten und später wieder zurückschreiben oder S-Merker verwenden, die in genügender Anzahl verfügbar sind.

Die Zeit wird in den Systemdaten BS 97 bis BS 100 oder im DB 1 vorgegeben.

| OB | Systemdatum | Priorität |
|---|---|---|
| 13 | BS 97 | Höchste |
| 12 | BS 98 | |
| 11 | BS 99 | |
| 10 | BS 100 | Niedrigste |

Die Zeitgesteuerte Programmbearbeitung ist durch einen Alarm (OB 2) oder eine zeitdauergesteuerte Programmbearbeitung (OB 6) unterbrechbar.

## Priorisierung

Die zeitdauergesteuerte und die alarmgesteuerte Programmbearbeitung hat immer Vorrang vor einer zeitgesteuerten Programmbearbeitung. Falls eine zeitdauergesteuerte Programmbearbeitung (OB 6) und eine Alarmbearbeitung (OB 2) gleichzeitig anstehen, so wird im normalen Fall zuerst der zeitdauergesteuerte Programmteil (OB 6) bearbeitet. Diese Priorisierung lässt sich durch setzen des Systemdatenbits BS 120.6 oder durch die Einstellung im DB 1 umkehren.

## Weckfehler

Ein Weckfehler wird gemeldet, wenn ein zeitdauergesteuertes oder zeitgesteuertes Programm bearbeitet wird und die Bearbeitung wird ein zweites Mal angefordert. Der Weckfehler wird ebenfalls ausgelöst falls die Bearbeitung während einer Alarmsperre ein zweites mal angefordert wird. Ein Weckfehler hat den Aufruf des OB 33 zur Folge. Im Akkumulator 1 wird ein Fehlercode übergeben, der den Weckfehler näher spezifiziert:

| Fehlercode | Ursache |
|---|---|
| 1 | Bearbeitung des OB 6 |
| 2 | Bearbeitung des OB 13 |
| 3 | Bearbeitung des OB 12 |
| 4 | Bearbeitung des OB 11 |
| 5 | Bearbeitung des OB 10 |

**Hinweis!**

Nähere Erläuterungen zur Alarmbearbeitung finden Sie in Teil "Alarm- und Zeitgesteuerte Bearbeitung".

# Teil 8 Einführung in die Programmiersprache

**Überblick**

Dieses Kapitel beschreibt die Programmierung von Automatisierungsaufgaben. Es wird erklärt, wie man Programme erstellt und welche Bausteine zur Gliederung eines Programms eingesetzt werden können. Außerdem finden Sie eine Übersicht der verschiedenen Zahlendarstellungsarten, die die Programmiersprache kennt.

**Inhalt**


| Thema | Seite |
|---|---|
| **Teil 8 Einführung in die Programmiersprache** | **8-1** |
| Vorgehensweise bei der Programmierung | 8-2 |
| Erstellung eines Programms | 8-4 |
| Programmstruktur | 8-8 |
| Bausteinarten | 8-9 |
| Programmbearbeitung | 8-22 |
| Bearbeiten von Bausteinen | 8-32 |
| Zahlendarstellung | 8-34 |
| Fehlerdiagnose im Programm | 8-35 |

# Vorgehensweise bei der Programmierung

Ein Steuerungsprogramm lässt sich in drei Abschnitte einteilen:

- Technologische Aufgabenstellung,
- Programmentwurf,
- Programmierung, Test und Inbetriebnahme.

**technologische Aufgabenstellung**

- Erstellen Sie ein grobes Blockdiagramm über die Steuerungsaufgaben Ihres Prozesses.
- Erstellen Sie eine Liste der für die Aufgabe benötigten Ein- und Ausgabesignale.
- Verfeinern Sie das Blockdiagramm, indem Sie den einzelnen Blöcken die Signale und evtl. erforderliche Zeitbedingungen bzw. Zählerstände zuordnen.

**Programm entwerfen**

- Entwerfen Sie die Bearbeitungsarten (zyklisch bzw. zeitgesteuert) Ihres Programms und benennen Sie die hierzu verwendeten OBs.
- Teilen Sie die Bearbeitungsarten in technologische bzw. funktionale Blöcke ein.
- Prüfen Sie, ob Sie die Blöcke einem Programm- oder Funktionsbaustein zuordnen können und benennen Sie die zu verwendenden Bausteine (PB x, FB y usw.)
- Klären Sie den Speicherbedarf für Zähler, Zeiten, Daten- und Ergebnisspeicher.
- Legen Sie die Aufgaben für jeden vorgesehenen Codebaustein und die Daten für evtl. benötigte Merker und Datenbausteine fest.
- Zeichnen Sie für die Codebausteine Ablaufdiagramme.

**Hinweis!**

Der Zyklus muss ausreichend schnell ablaufen. Die Prozesszustände dürfen sich nicht schneller ändern als die CPU darauf reagieren kann. Ansonsten könnte der Prozess außer Kontrolle geraten. Berücksichtigen Sie als maximale Reaktionszeit die doppelte Zykluszeit.

Sie bestimmen die Zykluszeit durch die zyklische Bearbeitung des Systemprogramms und durch Art und Umfang des Anwenderprogramms.

**Programmierung, Test und Inbetriebnahme**

- Legen Sie die Darstellungsart für die Codebausteine fest (FUP, KOP oder AWL).
- Programmieren Sie die Code- und Datenbausteine.
- Nehmen Sie die Bausteine nacheinander in Betrieb.

Wenn Sie sicher sind, dass alle Codebausteine richtig ablaufen und alle Daten richtig berechnet und abgespeichert werden, können Sie Ihr gesamtes Programm in Betrieb nehmen.

**Hinweis!**

Bitte beachten Sie, dass Sie die Funktionsbausteine nur in der Darstellungsart AWL erstellen können.

# Erstellung eines Programms

Bei speicherprogrammierbaren Steuerungen (SPS) werden Automatisierungsaufgaben in Form von Steuerungsprogrammen formuliert. Hier legt der Anwender in einer Reihe von Anweisungen fest, wie das AG die Anlage steuern oder regeln soll. Damit das Automatisierungsgerät (AG) das Programm "verstehen" kann, muss es in einer ganz bestimmten Sprache, der Programmiersprache, nach festgelegten Regeln geschrieben sein. So ist die Programmiersprache STEP 5 von Siemens entwickelt worden.

## Darstellungsarten KOP, FUP, AWL

Beim Programmieren können Sie für jeden einzelnen Code-Baustein zwischen den drei Darstellungsarten Kontaktplan (KOP), Funktionsplan (FUP) und Anweisungsliste (AWL) wählen, so dass die Programmiermethode dem jeweiligen Anwendungsfall angepasst werden kann.

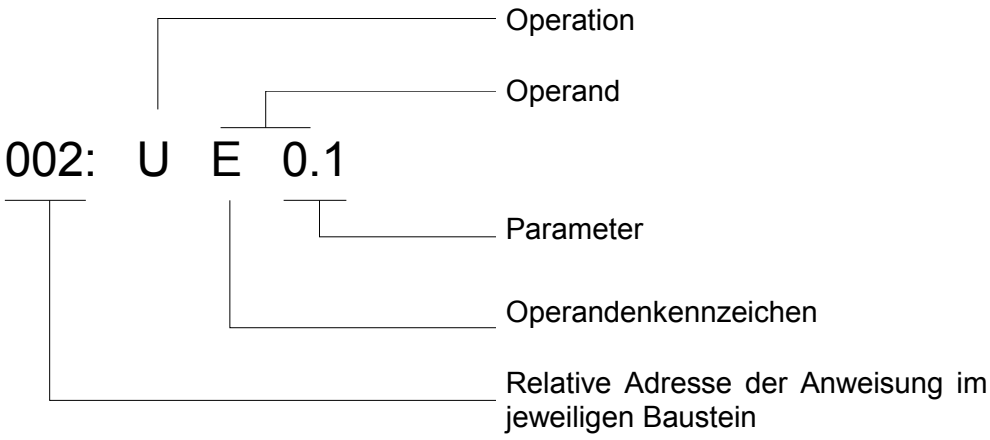
Der von den Programmiergeräten (PG) erzeugte Maschinencode MC 5 ist bei den drei Darstellungsarten identisch.

Wenn Sie beim Programmieren bestimmte Regeln berücksichtigen, kann das PG Ihr Anwenderprogramm von einer Darstellungsart in jede andere übersetzen!

### AWL

*Anweisungsliste*

Die AWL stellt das Programm als Abfolge von Befehlsabkürzungen dar. Eine Anweisung ist folgendermaßen aufgebaut.




Die Operation sagt dem AG, was es mit dem Operanden tun soll. Der Parameter gibt die Adresse eines Operanden an.

**FUP** *Funktionsplan*

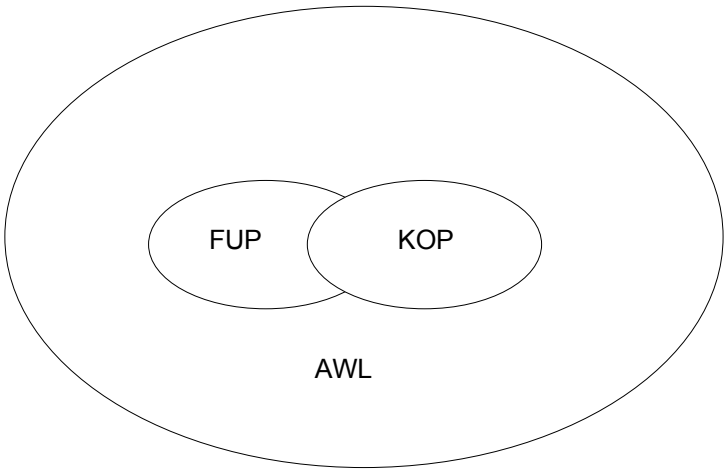
Im FUP werden die logischen Verknüpfungen mit Symbolen graphisch dargestellt.

**KOP** *Kontaktplan*

Im KOP werden die Steuerungsfunktionen mit Symbolen des Stromlaufplans graphisch dargestellt.

**Kompatibilität** Jede Darstellungsart besitzt besondere Eigenschaften. Ein Programmbaustein, der in AWL programmiert wurde, kann deshalb nicht ohne weiteres in FUP oder KOP ausgegeben werden. Auch die graphischen Darstellungsarten sind untereinander nicht kompatibel. Programme in FUP oder KOP können jedoch immer in AWL übersetzt werden.

Das folgende Bild stellt diese Aussagen in einem Mengendiagramm dar.



**Operationsarten**

Die Programmiersprache unterscheidet drei Arten von Operationen:

- Grundoperationen
- ergänzende Operationen
- Systemoperationen

| Programmiersprache | | | |
|---|---|---|---|
| | Grundoperationen | ergänzende Operationen | System-operationen |
| Anwendungs-bereich | in allen Bausteinen | nur im FB | nur im FB |
| Darstellungsarten | AWL, FUP, KOP | AWL | AWL |
| Besonderheiten | | | für Anwender mit guten System-kenntnissen |

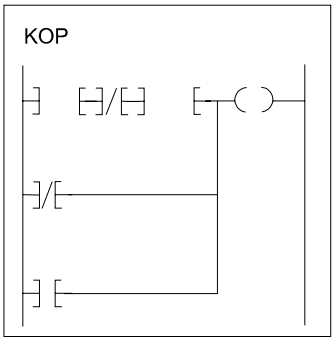
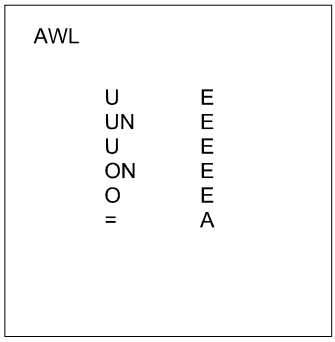
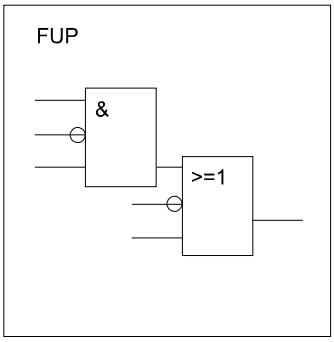
**Operanden-bereiche**

Die Programmiersprache kennt folgende Operandenbereiche:

| | | |
|---|---|---|
| E | (Eingänge) | Schnittstellen vom Prozess zum AG |
| A | (Ausgänge) | Schnittstellen vom AG zum Prozess |
| M | (Merker) | Speicher für binäre Zwischenergebnisse |
| S | (Merker) | Speicher für binäre Zwischenergebnisse |
| D | (Daten) | Speicher für digitale Zwischenergebnisse |
| T | (Zeiten) | Speicher zur Realisierung von Zeiten |
| Z | (Zähler) | Speicher zur Realisierung von Zählern |
| P | (Peripherie) | Schnittstelle zwischen Prozess und AG |
| K | (Konstanten) | Festgelegte Zahlenwerte |
| OB, PB, SB, FB, DB | (Bausteine) | Hilfsmittel zur Strukturierung des Programms |

### Graphische Darstellungen oder Liste mit Anweisungen

Während Sie mit den Darstellungsarten FUP und KOP die Möglichkeit haben, Ihr Programm graphisch darzustellen, werden in der Anweisungsliste die einzelnen Befehle aufgelistet.

| Kontaktplan | Anweisungsliste | Funktionsplan |
|---|---|---|
| Programmieren mit grafischen Symbolen wie Stromlaufplan<br><br>entspricht<br>DIN 19239 | Programmieren mit mnemotechnischen Abkürzungen der Funktionsbezeichnungen<br>entspricht<br>DIN 19239 | Programmieren mit graphischen Symbolen<br><br>entspricht<br>IEC 117-15<br>DIN 40700<br>DIN 40719<br>DIN 19239 |
|  |  |  |

# Programmstruktur

Beim der CPU 24x kann ein Programm linear oder strukturiert aufgebaut werden.

Die folgende Abschnitte beschreiben diese Programmformen.

## Lineare Programmierung

Zur Bearbeitung einfacher Automatisierungsaufgaben genügt es, die einzelnen Operationen in einem Abschnitt (Baustein) zu programmieren.

Beim Automatisierungsgerät ist dies der Organisationsbaustein 1, kurz "OB1". Dieser Baustein wird zyklisch bearbeitet, d. h. nach der letzten Anweisung wird wieder die erste Anweisung bearbeitet.

Dabei ist zu beachten:

- Beim Aufruf des OB 1 werden fünf Wörter für den Bausteinkopf belegt.
- Eine Anweisung belegt normalerweise ein Wort im Programmspeicher. Daneben gibt es auch 2-Wort-Anweisungen, z.B. mit den Operationen "Laden einer Konstanten". Sie müssen bei der Berechnung der Programmlänge doppelt gezählt werden.
- Der OB 1 muss - wie alle Bausteine - durch die Anweisung "BE" beendet werden.

## Strukturierte Programmierung

Zur Lösung komplexerer Aufgaben unterteilt man das Gesamtprogramm sinnvollerweise in einzelne, in sich abgeschlossene Programmteile (Bausteine).

Dieses Verfahren bietet Ihnen folgende Vorteile:

- einfache und übersichtliche Programmierung auch großer Programme,
- Möglichkeit zum Standardisieren von Programmteilen,
- leichte Änderungsmöglichkeiten,
- einfache Programmtests,
- einfache Inbetriebnahme,
- Unterprogrammtechnik (Aufruf eines Bausteins von verschiedenen Stellen aus).

# Bausteinarten

**Kennzeichnung**

Ein Baustein ist gekennzeichnet durch:

- den Bausteintyp (OB, PB, SB, FB, DB),
- die Bausteinnummer (Zahl zwischen 0 und 255).

Die wichtigsten Eigenschaften der einzelnen Bausteinarten finden Sie in der folgenden Tabelle:

| | OB | PB | SB | FB | DB |
|---|---|---|---|---|---|
| Anzahl | 256 *1<br>OB 0...OB 255 | 256<br>PB 0...PB 255 | 256<br>SB 0...SB 255 | 256 *2<br>FB 0...FB 255 | 254 *3<br>DB 2...DB255 |
| Länge (max.) | 8 x $2^{10}$ Byte | 8 x $2^{10}$ Byte | 8 x $^{10}$ Byte | 8 x $2^{10}$ Byte | 2042 Datenwörter *4 |
| Operationsvorrat (Inhalt) | Grundoperationen | Grundoperationen | Grundoperationen | Grundoperationen, ergänzende Operationen, Systemoperationen | Bitmuster<br>Zahlen<br>Texte |
| Darstellungsarten | AWL, FUP, KOP | AWL, FUP, KOP | AWL, FUP, KOP | AWL | |
| Bausteinkopflänge | 5 Wörter | 5 Wörter | 5 Wörter | 5 Wörter | 5 Wörter |

*1 Im Betriebssystem sind bereits Organisationsbausteine integriert. Einige OBs werden vom Betriebssystem aufgerufen.

*2 Im Betriebssystem sind bereits FBs integriert.

*3 Die Datenbausteine DB 0 und DB 1 sind reserviert.

*4 Bis DW 255 ansprechbar mit "L DW, L DL, L DR" oder "T DW, T DR, TDL" oder "P D,PN D, SU D, RU D".

**Baustein-Typen**

Die Programmiersprache unterscheidet folgende Bausteintypen:

***Organisationsbausteine (OB):***

Die Organisationsbausteine sind die Schnittstelle zwischen dem Systemprogramm und dem Anwenderprogramm. Sie verwalten das Steuerungsprogramm:

***Programmbausteine (PB):***

Programmbausteine werden zur Strukturierung des Anwenderprogramms verwendet und enthalten die nach technologischen oder funktionellen Gesichtspunkten gegliederten Teilprogramme. Die PB bilden den Kern des Anwenderprogramms.

***Schrittbausteine (SB):***

Schrittbausteine waren ursprünglich spezielle Programmbausteine zur schrittweisen Bearbeitung von Ablaufketten. Ablaufketten können jedoch inzwischen über GRAPH 5 (von Siemens) programmiert werden. Daher haben Schrittbausteine nicht mehr ihre ursprüngliche Bedeutung. Schrittbausteine sind jetzt eine zahlenmäßige Erweiterung der Programmbausteine. Sie können wie diese eingesetzt werden.

***Funktionsbausteine (FB):***

Funktionsbausteine dienen zum Programmieren von häufig wiederkehrenden oder auch von komplexen Funktionen (z.B. digitale Funktionen, Ablaufsteuerungen, Regelungen, Meldefunktionen). Ein Funktionsbaustein kann von übergeordneten Bausteinen mehrfach aufgerufen werden und bei jedem Aufruf mit neuen Operanden versorgt ("parametriert") werden.

***Datenbausteine (DB):***

In Datenbausteinen stehen die (festen oder veränderbaren) Daten, mit denen das Anwenderprogramm arbeitet. Diese Bausteinart enthält keine Anweisungen und unterscheidet sich in ihrer Funktion grundsätzlich von den übrigen Bausteinen.

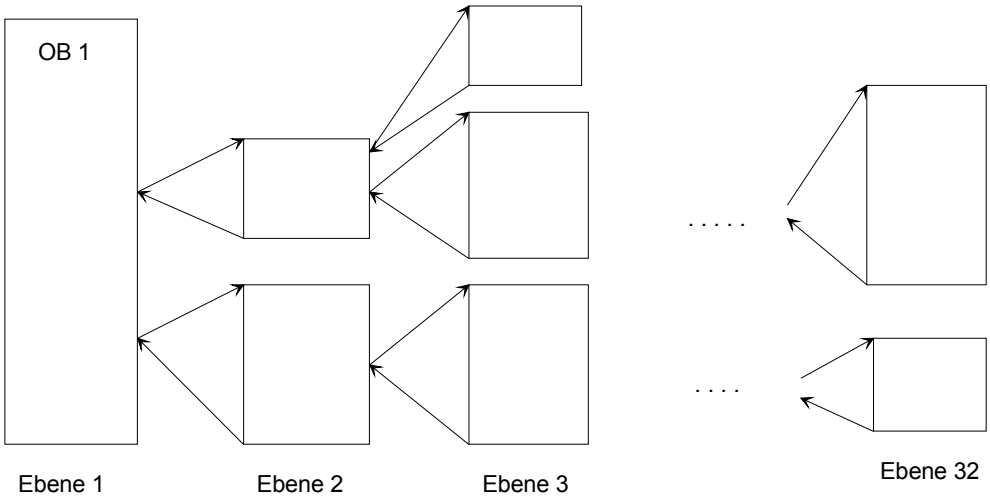
**Ebenen**

Mit Bausteinaufrufen kann ein Baustein verlassen und in einen anderen Baustein gesprungen werden. So können beliebig Programm-, Funktions- und Schrittbausteine in bis zu 32 Ebenen verschachtelt werden.

**Hinweis!**

Bei der Berechnung der Schachtelungstiefe ist zu berücksichtigen, dass das Systemprogramm bei bestimmten Ereignissen einen Organisationsbaustein selbständig aufrufen kann (z.B. OB 32).

Die Gesamtschachtelungstiefe ergibt sich als Summe der Schachtelungstiefen aller programmierten Bausteine. Bei einer Verschachtelung in mehr als 32 Ebenen geht das AG mit der Fehlermeldung "Bausteinstack-Überlauf" (STUEB) in den STOP-Zustand.



**Aufbau der Bausteine**

Alle Bausteintypen bestehen aus

- Bausteinkopf
- Bausteinrumpf.

***Bausteinkopf***

Der Bausteinkopf hat immer eine Länge von 5 Wörtern und enthält Informationen für die Bausteinverwaltung im PG und Daten für das Systemprogramm.

***Bausteinrumpf***

Im Bausteinrumpf sind abhängig vom Bausteintyp enthalten:

- Befehle (bei OB, PB, SB, FB),
- variable oder konstante Daten (bei DB)
- Formaloperandenliste (bei FB).

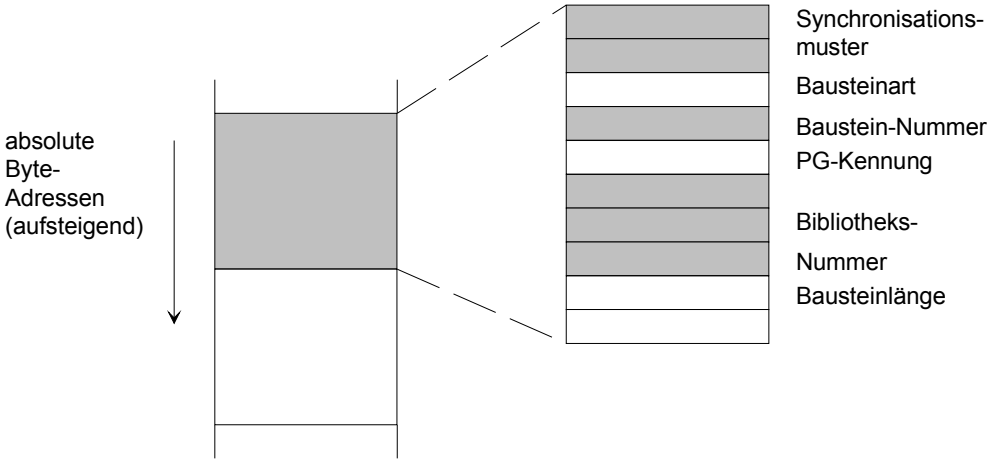
***Bausteinvorkopf***

Zu den Bausteinen vom Typ DB, FB erzeugt das Programmiergerät zusätzlich einen Bausteinvorkopf (DV, FV,). Diese Bausteinvorköpfe enthalten Informationen über das Datenformat (bei DB) bzw. über die Sprungmarken (bei FB), die nur das Programmiergerät auswerten kann. Die Bausteinvorköpfe werden deshalb nicht in den CPU-Speicher übertragen.

Als Anwender haben Sie auf den Inhalt eines Bausteinvorkopfes direkt keinen Einfluss.

***Maximale Länge***

Ein Baustein darf maximal 4096 Wörter (1 Wort entspricht 16 Bit) im Programmspeicher der CPU belegen.



**OB**

*Organisationsbausteine*

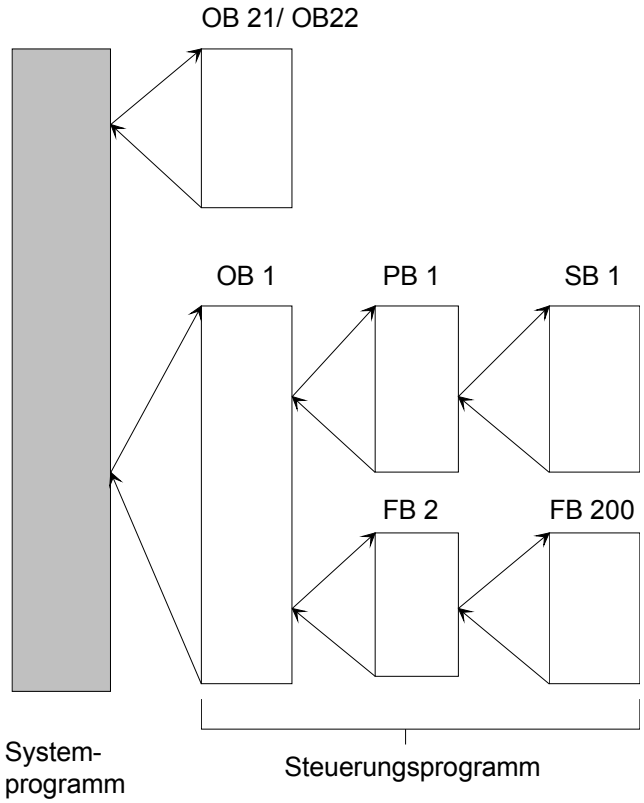
Organisationsbausteine bilden die Schnittstelle zwischen Betriebssystem und sie lassen sich in drei Gruppen zusammenfassen:

- Ein OB wird zyklisch vom Betriebssystem aufgerufen (OB 1).
- Ein Teil der OBs ist ereignis- oder zeitgesteuert, d.h. sie werden aufgerufen durch.
  - STOP→RUN - bzw. Netz AUS→NETZ-EIN-Übergang (OB 21, OB 22) Alarme (OB 2, OB 6).
  - Programmier- bzw. Gerätefehler (OB 19, OB 23, OB 24, OB 27, OB 32, OB 33, OB 34).
  - Ablauf eines Zeitintervalls (OB 10...OB 13).
- Ein anderer Teil stellt Betriebsfunktionen dar (ähnlich den integrierten FBs) die vom Steuerungsprogramm aufgerufen werden können.

**System-OBs**

| OB-Nr | Funktion |
|---|---|
| **OB wird zyklisch vom Betriebssystem aufgerufen** | |
| OB 1 | zyklische Programmbearbeitung |
| **OBs für alarm- und zeitgesteuerte Programmbearbeitung** | |
| OB 2 | Alarm: Alarmgenerierung durch die Digital- Eingabebaugruppe |
| OB 6 | Alarm ausgelöst durch internen Timer |
| OB 10 | zeitgesteuerte Programmbearbeitung (jeweils variabel: 10ms...10min) |
| OB 11 | zeitgesteuerte Programmbearbeitung (jeweils variabel: 10ms...10min) |
| OB 12 | zeitgesteuerte Programmbearbeitung (jeweils variabel: 10ms...10min) |
| OB 13 | zeitgesteuerte Programmbearbeitung (jeweils variabel: 10ms...10min) |
| **OBs für die Steuerung des Anlaufverhaltens** | |
| OB 21 | bei manuellem Einschalten (STOP→RUN) |
| OB 22 | bei Spannungswiederkehr |
| **OBs für die Behandlung von Programmier- und Gerätefehlern** | |
| OB 19 | bei Aufruf eines nicht geladenen Bausteins |
| OB 23 | Quittungsverzug bei Einzelzugriffen auf den Rückwandbus (z.B. L PB, LIR usw.) |
| OB 24 | Quittungsverzug beim Aktualisieren des Prozessabbilds |
| OB 27 | Substitutionsfehler |
| OB 32 | Transferfehler im DB oder bei EDB-Operation |
| OB 33 | Weckfehler |
| OB 34 | Batterieausfall |
| **OBs, die Betriebsfunktionen bereitstellen** | |
| OB 31 | Zykluszeittriggerung |
| OB160 | programmierbare Zeitschleife |
| OB251 | PID-Regelalgorithmus |

Das folgende Bild zeigt, wie die Organisationbausteine vom Systemprogramm aufgerufen werden. Es verdeutlicht die Bedeutung von OBs und zeigt wie Sie ein strukturiertes Steuerungsprogramm aufbauen können.



**PB** *Programmbausteine*

In diesen Bausteinen werden normalerweise abgeschlossene Programmteile mit dem Befehlsvorrat der Grundoperationen programmiert.

Besonderheit:

Steuerungsfunktionen lassen sich in Programmbausteinen graphisch darstellen.

Aufruf:

PBs werden durch die Bausteinaufrufe SPA und SPB aktiviert. Diese Operationen können, außer in Datenbausteinen, in allen Bausteintypen programmiert werden. Bausteinaufruf und -ende begrenzen das VKE. Es kann jedoch in den "neuen " Baustein mitgenommen und ausgewertet werden.

**SB** *Schrittbausteine*

Schrittbausteine sind Sonderformen von Programmbausteinen zur Bearbeitungen von Ablaufsteuerungen. Sie werden wie Programmbausteine behandelt.

**FB** *Funktionsbausteine*

In Funktionsbausteinen werden häufig wiederkehrende oder komplexe Steuerungsfunktionen programmiert.

Besonderheiten:

- FBs lassen sich parametrieren
  Beim Bausteinaufruf können Aktualparameter übergeben werden.
- Gegenüber anderen Bausteinen steht ein erweiterter Operationsvorrat zur Verfügung.
- Das Programm lässt sich nur als AWL erstellen und dokumentieren,

Beim Automatisierungsgerät gibt es verschiedene Ausführungen von FBs; sie sind:

- vom Anwender programmierbar,
- im Betriebssystem integriert oder,
- als Softwarepakete (Standard-FBs) erhältlich.

**Bausteinkopf**

FBs besitzen, im Gegensatz zu anderen Bausteinarten, zusätzlich zum Bausteinkopf noch andere Organisationsinformationen.

Der Speicherbedarf für Organisationsinformationen ergibt sich aus:

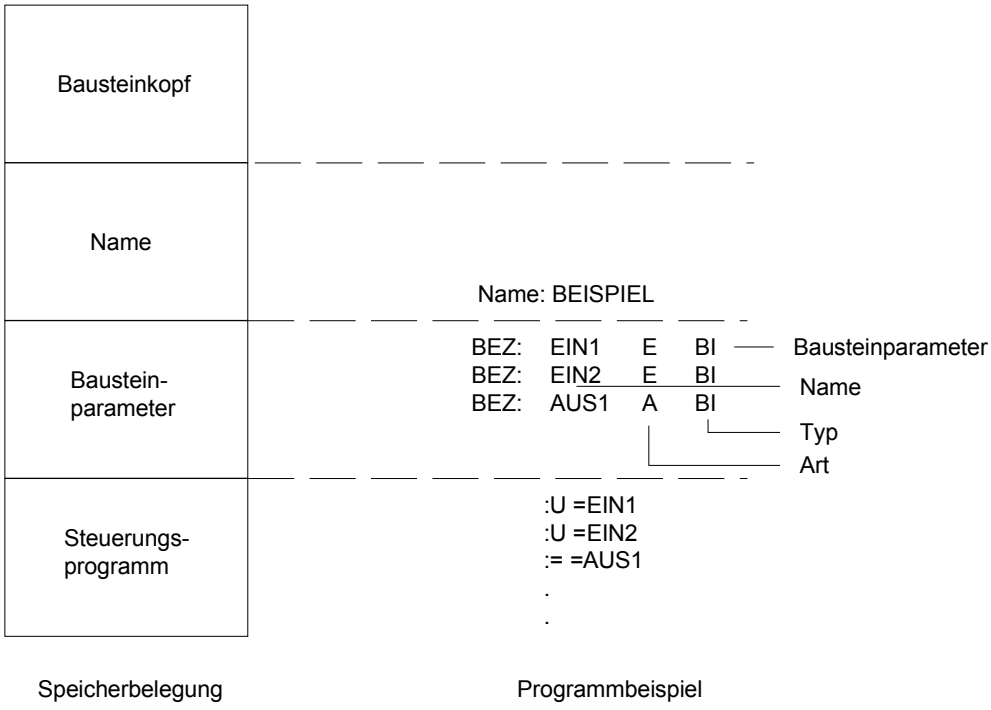
- Bausteinkopf wie bisher 5 Wörter,
- Bausteinname (5 Wörter),
- Bausteinparameter bei Parametrierung (3 Wörter je Parameter).

**Erstellung**

Im Gegensatz zu anderen Bausteinen können FBs parametriert werden. Für die Parametrierung müssen Sie folgende Angaben über die Bausteinparameter programmieren:

- **Namen** der Bausteinparameter (Formaloperanden)
  Jeder Formaloperand erhält einen Namen (Bezeichnung "BEZ"). Der Name darf höchstens aus vier Zeichen bestehen und muss mit einem Buchstaben beginnen. Pro FB können Sie bis zu 40 Bausteinparameter programmieren.
- **Art** des Bausteinparameters
  Folgende Parameterarten können eingegeben werden:
  E Eingangsparameter
  A Ausgangsparameter
  D Datum
  B Baustein
  T Zeit
  Z Zähler
- Typ des Bausteinparameters
  Sie können folgende Typen angeben:
  BI Für Operanden mit Bitadresse
  By für Operanden mit Byteadresse
  W für Operanden mit Wortadresse
  K für konstante Werte

Bei Parametrierung müssen Name, Art und Typ des Bausteinparameters eingegeben werden.



**Parameter**

| Parameter | Typ des Parameters | Zugelassene Aktualoperanden |
|---|---|---|
| E, A | BI für einen Operanden mit Bitadresse<br><br>BY für einen Operanden mit Byteadr. | E x,y Eingänge<br>A x,y Ausgänge<br>M x,y Merker<br>EB x Eingangsbytes<br>AB x Ausgangsbytes<br>MB x Merkerbytes<br>DL x Datenbytes links<br>DR x Datenbytes rechts<br>PB x Peripheriebytes |
| D | KM für ein Binärmuster (16 Stellen)<br>KY für zwei byteweise Betragszahlen im Bereich jeweils von 0 bis 255<br>KH für ein Hexadezimalmuster (max.4 Stellen<br>KC für ein Zeichen (max. 2 alpha-numerischeZeichen)<br>KT für einen Zeitwert (BCD-codierter Zeitwert) mit Zeitraster 1.0 bis 999.3<br>KZ für einen Zählerwert (BCD-codiert)<br>0bis 999<br>KF für eine Festpunktzahl im Bereich von -32768 bis +32767 | Konstanten |
| B | keine Typanzeige zulässig | DB x DB, ausgeführt wird der Befehl ADBx.<br>FB x FB (nur ohne Parameter zulässig) werden absolut (SPA..x) aufgerufen.<br>PB x PB werden absolut (SPA..x) aufgerufen<br>SB x SB werden absolut (SPA..x) aufgerufen<br>OB x OB werden absolut (SPA..x) aufgerufen |
| T | keine Typanzeige zulässig | T Zeit; der Zeitwert ist als Datum zu parametrieren oder als Konstante im FB zu programmieren. |
| Z | keine Typanzeige zulässig | Z Zähler; der Zählwert ist als Datum zu parametrieren oder als Konstante im FB zu programmieren. |

**FB aufrufen**

FBs werden wie die anderen Bausteine unter einer bestimmten Nummer (z.B. FB 47) im internen Programmspeicher abgelegt. Die Nummern 238...251 sind für integrierte FBs reserviert und können daher nicht für "selbstgeschriebene" FBs genutzt werden! Die FBs 238 und 239 sind ebenfalls für integrierte FBs reserviert; diese FBs können aber umnummeriert werden.

In allen Bausteinen, außer den DBs, können Aufrufe von FBs programmiert werden.

Der Aufruf eines Funktionsbausteins setzt sich zusammen aus:

- Aufrufanweisung
  SPA FBx absoluter Aufruf des FB x (**SP**ringe **A**bsolut...)
  SPB FBx Aufruf des FB x, nur wenn VKE =1 (**SP**ringe **B**edingt...)
- Parameterliste (nur notwendig, wenn im FB Bausteinparameter definiert wurden).

**FB parametrieren**

Das Programm im Funktionsbaustein legt fest, wie die Formaloperanden (das sind die Parameter, die als "BEZ" definiert wurden) bearbeitet werden sollen.

Sobald Sie die Aufrufanweisung (z.B. SPA FB 2) programmiert haben blendet das PG die Parameterliste ein. Die Parameterliste besteht aus den Namen der Parameter, jeweils gefolgt von einem Doppelpunkt (:). Den Parametern müssen nun sogenannte Aktualoperanden zugeordnet werden. Aktualoperanden ersetzen beim Aufruf des FBs die dort definierten Formaloperanden, so dass der FB "eigentlich" mit den Aktualoperanden arbeitet.

Die Parameterliste darf max. 40 Parameter umfassen.

Beispiel:

Der Name (BEZ) eines Parameters ist EIN1, die Art ist E (wie Eingang) und der Typ ist BI (wie Bit).

Der Formaloperand des FBs hat dann die Form:

```
BEZ:      EIN1    E   BI
```

Im aufrufenden Baustein wird in der Parameterliste festgelegt, welcher (Aktual-) Operand im Falle des FB-Aufrufs den Formaloperanden ersetzen soll; im Beispiel soll dies der Operand "E 1.0" sein. In der Parameterliste muss folglich eingetragen werden:

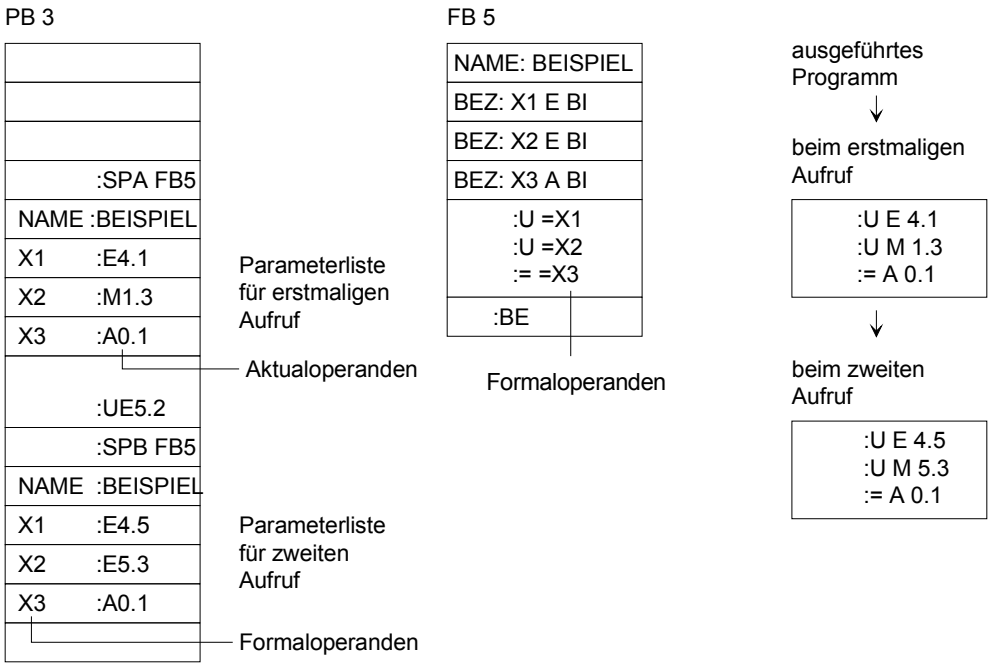
```
EIN1:     E 1.0
```

Wenn der FB aufgerufen wird, setzt er an Stelle des Formaloperanden "EIN1" den Aktualoperanden "E1.0".

Der FB-Aufruf belegt im internen Programmspeicher zwei Wörter, jeder Parameter ein weiteres Speicherwort.

Die bei der Programmierung am Programmiergerät erscheinenden Bezeichner für die Ein- und Ausgänge des FBs, sowie der Name, sind im FB selbst abgelegt.

Deshalb müssen, bevor mit der Programmierung am Programmiergerät begonnen wird, alle erforderlichen FBs auf die Programmdiskette übertragen (bei Off-line-Programmierung) oder direkt in den Programmspeicher des AGs eingegeben werden.



**DB**

*Datenbausteine*

In Datenbausteinen legen Sie Daten ab, die im Programm bearbeitet werden sollen.

Folgende Arten von Daten sind zulässig:

- Bitmuster (Darstellung von Anlagenzuständen),
- Zahlen in Hexa-, Dual- oder Dezimal-Schreibweise (Zeitwerte, Rechenergebnisse),
- alphanumerische Zeichen (Meldetexte).

**Programmierung von DBs**

Die Programmierung eines DBs beginnt mit der Angabe einer Baustein-Nummer zwischen 2 und 255. Reserviert sind DB 0 (für das Betriebssystem) und DB 1 (für die Parametrierung interner Funktionen).

Die Daten werden wortweise in diesem Baustein abgelegt. Umfasst die Information weniger als 16 Bit, so werden die höherwertigen Bits mit Nullen aufgefüllt. Die Eingabe von Daten beginnt beim Datenwort 0 und wird in aufsteigender Reihenfolge fortgesetzt. Ein DB kann bis zu 2042 Datenwörter aufnehmen. Bis DW 255 kann mit den Befehlen "L DW" und "T DW" zugegriffen werden. Ein Zugriff auf die Datenworte 256...2042 ist nur möglich mit den Operationen "LIR", "TIR" und "TNB".

| Eingabe | | | | | gespeicherte Werte | |
|---|---|---|---|---|---|---|
| 0000 | : | KH | = | A13C | DW0 | A13C |
| 0001 | : | KT | = | 100.2 | DW1 | 2100 |
| 0003 | : | KF | = | +21874 | DW2 | 5572 |

DBs können auch im Steuerungsprogramm erzeugt oder gelöscht werden.

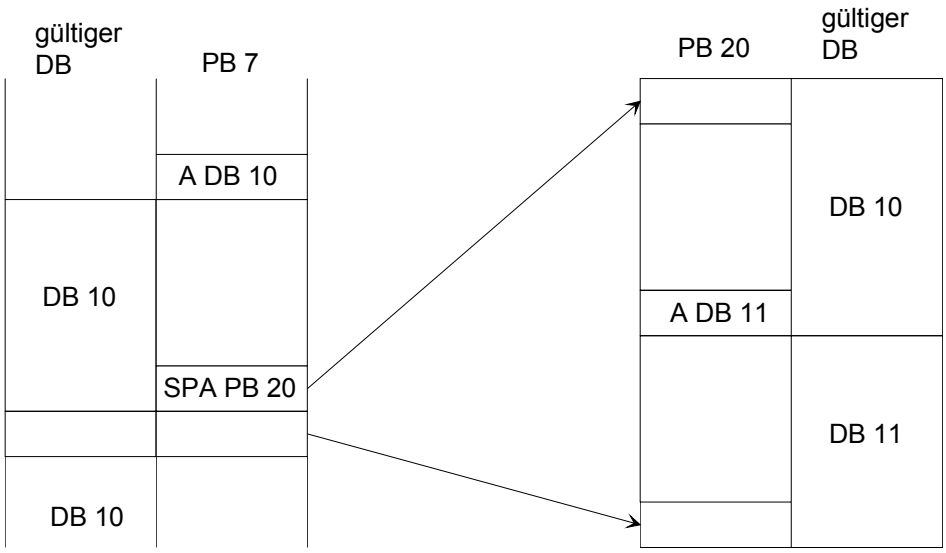
**Programm-bearbeitung mit DBs**

Ein DB muss im Programm mit dem Befehl A DBx (x=2...255) aufgerufen werden.

Ein DB bleibt - innerhalb eines Bausteines - so lange gültig, bis ein anderer DB aufgerufen wird.

Beim Rücksprung in den übergeordneten Baustein gilt wieder der DB, der vor dem Bausteinaufruf gültig war.

In allen Organisationsbausteinen (OBs) müssen die vom Anwender-programm benutzten DBs mit einem entsprechenden A DB x-Befehl aufgeschlagen werden.




Beim Aufruf des PB 20 wird der gültige Datenbereich in einen Speicher eingetragen.

Beim Rücksprung wird dieser Datenbaustein wieder aufgeschlagen.

# Programmbearbeitung

Ein Teil der Organisationsbausteine (OBs) übernimmt die Aufgabe der Strukturierung und Verwaltung des Steuerungsprogramms.

Diese OBs lassen sich nach folgenden Aufgaben gruppieren:

- OBs für ANLAUF-Programmbearbeitung,
- OB für die zyklische Programmbearbeitung,
- OBs für die zeitgesteuerte Programmbearbeitung,
- OBs für die (Prozess-) alarmgesteuerte Programmbearbeitung,
- OBs für die Behandlung von Programmier- und Gerätefehlern.

Daneben gibt es noch OBs, die Funktionen ähnlich den integrierten Funktionen bieten (z.B. PID-Regelalgorithmus)

## ANLAUF-Programm-bearbeitung

Im ANLAUF, d.h.

- nach einem STOP→RUN-Übergang (manueller Neustart)
- nach einem NETZ AUS→NETZ EIN-Übergang (automatischer Neustart nach Netzwiederkehr, wenn die CPU vorher in der Betriebsart RUN gewesen ist),

ruft das Betriebssystem der CPU automatisch (vorausgesetzt er ist programmiert) einen ANLAUF-OB auf:

- OB 21 (bei manuellem Neustart)

oder

- OB 22 (bei automatischem Neustart nach Netzwiederkehr, wenn die CPU vorher in der Betriebsart RUN gewesen ist).

Wenn Sie diese Bausteine programmiert haben, wird dieses Programm vor der zyklischen Programmbearbeitung abgearbeitet; es eignet sich somit z.B. für die (einmalige) Voreinstellung bestimmter Systemdaten.

Ist der entsprechende ANLAUF-OB nicht programmiert, verzweigt die CPU direkt in die Betriebsart RUN (zyklische Programmbearbeitung, OB 1).

**Beispiel 1**

Programmierung des OB 22:

Nach Netzwiederkehr soll überprüft werden, ob noch alle Ein- und Ausgabebaugruppen betriebsbereit sind. Wenn eine oder mehrere Baugruppen nicht mehr ansprechbar sind (nicht gesteckt oder fehlerhaft), dann soll das AG in die Betriebsart STOP verzweigen.

| OB 22 AWL | | | Erläuterung |
|---|---|---|---|
| :L | KF | +0 | Die Ausgangsworte 0, 2 und 4 |
| :T | PW | 0 | werden auf "0"gesetzt. |
| :T | PW | 2 | |
| :T | PW | 4 | |
| :L | PW | 6 | Die Informationen der Eingangsworte |
| :L | PW | 8 | 6, 8 und 10 werden nacheinander |
| :L | PW | 10 | in den AKKU 1 geladen. |
| :BE | | | |

Ist eine Ein- oder Ausgabebaugruppe mit der Anweisung LPW bzw. TPW nicht ansprechbar, geht die CPU bei dieser Anweisung in den Zustand STOP und das Unterbrechungsbit QVZ (Quittungsverzug) im USTACK wird gesetzt.

**Beispiel 2**

Programmierung des OB 21 und des FB 1:

Nach Neustart durch den Betriebsartenschalter sollen die Merkerbytes 0 bis 99 mit "0" vorbesetzt werden, die Merkerbytes 100 bis 127 sollen erhalten bleiben, da sie wichtige Maschineninformationen beinhalten.

Voraussetzung: Remanenzschalter auf Stellung (RE).

| OB 21 AWL | | | | Erläuterung |
|---|---|---|---|---|
| | :SPA | FB | 1 | Absoluter Aufruf des FB 1 |
| NAME | :LÖSCH | M | | |
| | :BE | | | |

| FB 1 AWL | | | | Erläuterung |
|---|---|---|---|---|
| NAME | LOES | CH | M | |
| | :L | KF | +0 | Das MW 200 wird mit "0"vorbesetzt |
| | :T | MW | 200 | Der Wert "0" wird im AKKU1 gesp. |
| M10 | :L | KF | +0 | Der Inhalt von MW 200 gibt die Adr. |
| | :B | MW | 200 | des aktuellen MW an |
| | :T | MW | 0 | Das akt. MW wird auf "0" gesetzt |
| | :L | MW | 200 | |
| | :I | 2 | | Der Inhalt von MW 200 wird um 2 erhöht |
| | :T | MW | 200 | |
| | :L | KF | +100 | Der Vergleichswert "100" wird in den AKKU 1 geladen |
| | :<F | | | Solange der Inhalt von MW 200 < 100 |
| | :SPB | =M10 | | ist, wird zu M10 gesprungen |
| | :BE | | | Die Bytes MB 0...99 sind auf "0" gesetzt |

### Zyklische Programmbearbeitung

Der OB 1 wird vom Betriebssystem zyklisch aufgerufen. Die maximale Zeitdauer des zyklischen Programms ist durch die Zyklus-Überwachungszeit festgelegt. Wenn Sie strukturiert programmieren wollen, sollten Sie im OB 1 nur Sprungoperationen (Bausteinaufrufe) programmieren. Die aufgerufenen Bausteine (PBs, FBs und SBs) sollten dann angeschlossene Funktionseinheiten enthalten, so dass die Übersichtlichkeit erhöht wird.

Ein Default OB 1 ist immer in der CPU!

### Zeitgesteuerte Programmbearbeitung

Für die zeitgesteuerte Programmbearbeitung stehen Ihnen die OBs 10...13 zur Verfügung. Die Zeit-OBs werden vom Betriebssystem in festgelegten Zeitintervallen aufgerufen.

Das Aufrufintervall kann (z.B. im ANLAUF-OB) in den Systemdaten als Vielfaches von 10ms eingestellt werden. Es kann während der zyklischen Programmbearbeitung verändert werden. Voreingestellt ist für den OB 13 ein Aufrufintervall von 100 ms. Aufrufintervalle von 10 ms bis 10 min sind in den Systemdatenworten 97...100 einstellbar (Bereich: 0...FFFFh)

Sie können die Aufrufintervalle auch im DB 1 parametrieren.

Das Betriebssystem ruft nur dann einen Zeit-OB auf,

- wenn ein Aufrufintervall >0 eingestellt ist

und

- wenn der entsprechende Zeit-OB programmiert ist.

Zeit-OBs unterbrechen das zyklisch Programm nach jeder Operation. Zeit-OBs können **nicht** unterbrechen:

- Integrierte Funktionsbausteine,
- den OB 6,
- Prozessalarme (OB 2).

Zeit-OBs ihrerseits können durch den OB 6 oder durch Prozessalarme (OB 2) unterbrochen werden! Beachten Sie, dass dadurch die Aufrufintervalle variiren können.

Die Prioritäten der Zeit-OBs untereinander sind wie folgt festgelegt:

| | |
|---|---|
| höchste Priorität: | OB 13 |
| | OB 12 |
| | OB 11 |
| niedrigste Priorität: | OB 10. |

**Bitte beachten**

Mit der Operation "AS" kann der Aufruf aller Zeit-OBs gesperrt, mit "AF" wieder freigegeben werden. Es kann eine Aufrufanforderung während einer Aufrufsperre gespeichert werden.

Wenn Zeit-OBs schon im ANLAUF-OB (OB 21, OB 22) bearbeitet werden sollen, dann müssen Sie im ANLAUF-OB Alarme mit "AF" freigeben.

Auch bei der Bearbeitung eines zeitgesteuerten OBs darf die Baustein-Schachtelungstiefe von 32 Ebenen nicht überschritten werden.

Werden von einem zeitgesteuerten OB "Schmiermerker" verwendet, die auch im zyklischen Steuerungsprogramm verwendet werden, müssen diese während der Zeit-OB-Bearbeitung in einen DB gerettet werden.

**Parameterblock für Zeit-OBs**

| System-datenwort | absolute Adresse | High-Byte | Low-Byte | Vorein-stellung |
|---|---|---|---|---|
| SD 97 | EAC2 | Zeitintervall für OB 13 (0...FFFFh •10ms) | | 10 (=100ms) |
| SD 98 | EAC4 | Zeitintervall für OB 12 (0...FFFFh•10ms) | | 0 (=kein Aufruf) |
| SD 99 | EAC6 | Zeitintervall für OB 11 (0...FFFFh •10ms) | | 0 (=kein Aufruf) |
| SD 100 | EAC8 | Zeitintervall für OB 10 (0...FFFFh •10ms) | | 0 (=kein Aufruf) |

| Einstellen einer Intervallzeit von 1s für den OB13: | | |
|---|---|---|
| OB 21<br>: SPA FB 21<br>NAME : ZEIT EIN<br>:<br>: | OB 22<br>: SPA FB 21<br>NAME : ZEIT EIN<br>:<br>: | FB 21<br><br>NAME : ZEIT EIN<br>: L KF +100<br>: T BS 97<br>: BE |

**Alarmgesteuerte Programmbearbeitung**

Eine Alarmbearbeitung liegt vor, wenn ein vom Prozess kommendes Signal die CPU veranlasst, die zyklische Programmbearbeitung zu unterbrechen und ein spezifisches Alarm-Programm zu bearbeiten.

Nach der Bearbeitung erfolgt ein Rücksprung an die Unterbrechungsstelle.

**(Prozess-) Alarm Reaktion mit OB 2**

Sobald ein (Prozess-) Alarm auftritt ruft das Betriebssystem den OB 2 auf. Sofern der OB 2 programmiert ist, wird dieser abgearbeitet. Danach kehrt das System wieder zurück an die Unterbrechungsstelle und setzt die Programmbearbeitung fort.

**Weck-Alarm-Reaktion mit OB 6**

Eine Sonderstellung nimmt der OB 6 ein. Der OB 6 wird vom Betriebssystem aufgerufen, wenn eine voreingestellte Zeit im Systemdatenwort 101 (EACAh) abgelaufen ist (vorausgesetzt, Alarme sind nicht durch die Operation "AS" gesperrt).

Im OB 6 programmieren Sie die Reaktion nach Ablauf der voreingestellten Zeit ("Weckalarm"). Die Zeit starten Sie durch einen Eintrag in das Systemdatenwort 101 (EACAh) ausschließlich mit dem Befehl T BS 101.

Beispiel:

Sie haben den OB 6 programmiert mit einer "Weckreaktion". Der OB 6 soll 22ms nach Starten der Weckzeit aufgerufen werden. Sie wählen und starten die "Weckzeit" durch die Operationen

```
L KF +22
T BS 101
```

Nach 22 ms unterbricht der OB 6 das laufende zyklische oder zeitgesteuerte Programm.

**Hinweis!**

Eine laufende "Weckzeit" kann "nachgetriggert" werden, indem Sie erneut einen Wert in das Systemdatenwort 101 transferieren. Das Betriebssystem startet dann erneut die Weckzeit, die durch den Wert in AKKU 1 vorgegeben wurde.

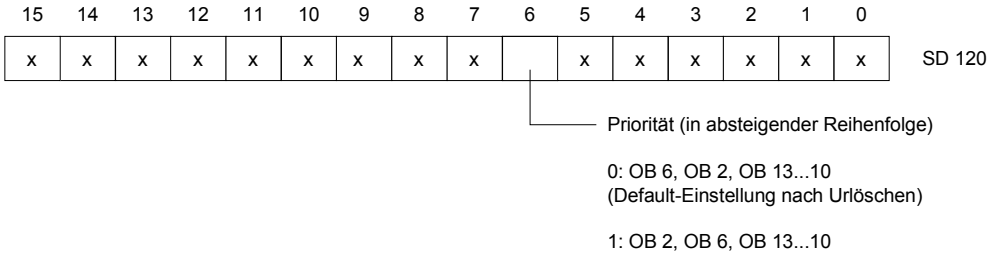
Eine laufende "Weckzeit" kann gestoppt werden (Aufruf des OB 6 wird verhindert!), indem Sie den Wert "0" in das Systemdatenwort 101 transferieren.

Nach dem Start der "Weckzeit" steht im Systemdatenwort 101 die gewählte Zeit bis zum Aufruf des OB 6. Wenn die programmierte Zeit abgelaufen ist, trägt das Betriebssystem den Wert "0" in das Systemdatenwort 101 ein und ruft den OB 6 auf.

**Für OB 6 gilt:**

Zum Starten der "Weckzeit" muss immer eine Zahl (Bereich1...65535 bzw. 1h...FFFFh) in das Systemdatenwort 101 (EACAh) transferiert werden.

- Die Weckzeit kann in 1ms-Schritten eingestellt werden, die einstellbare Weckzeit liegt damit im Bereich 1...65535ms.
- Die Priorität des OB 6 ist im DB 1 oder Systemdatenwort 120 einstellbar:


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
x x x x x x x x x x x x x x x
SD 120
Priorität (in absteigender Reihenfolge)
0: OB 6, OB 2, OB 13...10
(Default-Einstellung nach Urlöschen)
1: OB 2, OB 6, OB 13...10

x Systemeigenschaften bestimmende Bits
(dürfen bei der Einstellung der Priorität des OB 6 nicht verändert werden).

- Der OB 6 selbst ist nicht unterbrechbar.
- Der OB 6 kann das zyklische und zeitgesteuerte Programm, nicht aber ein laufendes Alarmprogramm (OB 2) unterbrechen! Wenn die "Weckzeit" abläuft während ein Alarm-OB bearbeitet wird, verzögert sich daher der Aufruf des OB 6.
- Der Aufruf des OB 6 kann sich außerdem verzögern, wenn:

– integrierte FBs verwendet werden,
– die integrierte Uhr parametriert ist,
– PG/OP-Funktionen anstehen,
– Rechnerkopplung oder ASCII-Treiber aktiviert sind

oder

– zeitgesteuerte OBs programmiert sind.

## Behandlung von Programmier- und Gerätefehlern

Mit den Fehlerreaktions-OBs ist es möglich, das Verhalten der CPU beim Auftreten von Fehlern weitgehend zu bestimmen.

Die Operation, die den Fehler Quittungsverzug, Substitutionsfehler oder Transferfehler auslöst, wird durch den Aufruf des zugehörigen Fehlerreaktions-OB ersetzt. in diesen OBs kann gezielt auf die Fehler reagiert werden. Ist dort nur "BE" programmiert, erfolgt keine Reaktion, d.h. das AG geht nicht in den STOP-Zustand. Wenn kein entsprechender OB vorhanden ist, verzweigt die CPU in die Betriebsart STOP.

### OB19

**Reaktion bei Aufruf eines nicht geladenen Bausteins**

Im OB 19 können Sie das Verhalten der CPU bei Aufruf eines nicht geladenen Bausteins programmieren.

Beispiel:

Die CPU soll bei Aufruf eines nicht geladenen Bausteins in den STOP-Zustand gehen.

| OB 19 AWL | Erläuterung |
|---|---|
| :STP<br>:BE | STOP-Anweisung |

Wenn der OB 19 nicht programmiert ist, fährt das Steuerungsprogramm unmittelbar nach der Sprunganweisung (mit nicht vorhandenem Sprungziel!) mit der Programmbearbeitung fort (keine Reaktion!).

### OB 23

**Reaktion auf Quittungsverzug bei Direktzugriff auf Peripherie**

Folgende Befehle können einen Quittungsverzug zur Folge haben: L PB; L PW; T PB; T PW; LIR; TIR; TNB.

Der Fehler Quittungsverzug (QVZ) tritt auf, wenn eine Baugruppe nach dem Ansprechen nicht innerhalb von 160 µs quittiert. Ursache kann ein Programmfehler, ein Defekt auf der Baugruppe oder das Entfernen der Baugruppe während der Betriebsart RUN sein.

Im Systemdatenwort 103 (EACEh) wird vom Betriebssystem die absolute Baugruppenadresse, bei der der QVZ auftrat, hinterlegt und der OB 23 aufgerufen. Wenn der OB 23 nicht vorhanden ist, geht die CPU mit "QVZ" in den STOP-Zustand.

### OB 24

**Reaktion auf Quittungsverzug beim Aktualisieren des Prozessabbildes**

Tritt beim Aktualisieren des Prozessabbildes ein Quittungsverzug auf, so wird im Systemdatenwort 103 (EACEh) die absolute Baugruppenadresse hinterlegt und der OB 24 aufgerufen. Wenn der OB 24 nicht vorhanden ist, geht die CPU mit "QVZ" in den STOP-Zustand.

**OB 27** **Reaktion auf Substitutionsfehler**

Ein Substitutionsfehler (SUF) kann auftreten, wenn in einem FB nach der Programmierung eines Aufrufes ("SPA FBx", "SPB FBx") seine Formalparameterbeschreibung geändert wird.

Das Betriebssystem unterbricht bei einem erkannten Substitutionsfehler das Steuerungsprogramm und bearbeitet statt der Substitutionsoperation den OB 27. Wenn der OB 27 nicht vorhanden ist, geht die CPU mit der USTACK-Fehlerkennung "SUF" in den STOP-Zustand.

**OB 32** **Reaktion auf Transferfehler**

Ein Transferfehler (TRAF) liegt vor, wenn:

- auf Datenwörter zugegriffen wurde, ohne vorher einen Datenbaustein aufzurufen (A DB),
- bei den Operationen L DW; T DW; P D; PN D; SU D; RU D; usw. der Parameter größer ist als die Länge des aufgeschlagenen Datenbausteins,
- beim Befehl E DB (Erzeuge Datenbaustein) der freie Anwenderspeicher nicht ausreicht, um den angegebenen Datenbaustein einzurichten.

Reaktion auf Transferfehler: Das Betriebssystem unterbricht die Abarbeitung der Operation, bei der ein Transferfehler auftrat, und bearbeitet statt dessen den OB 32. Wenn der OB 32 nicht vorhanden ist, geht die CPU mit der USTACK-Fehlerkennung "TRAF" in den STOP-Zustand.

**OB 33** **Weckfehler**

Ein Weckfehler wird gemeldet, wenn:

- einer der Organisationsbausteine OB 6 und OB 10 bis OB 13 bearbeitet wird und der die Bearbeitung auslösende Weckalarm bzw. Zeitdaueralarm kommt zum zweiten Mal,
- während einer Alarmsperre ein Weckalarm oder ein Zeitdaueralarm ein zweites Mal auftritt.

Bei einem Weckfehler wird der OB 33 aufgerufen. Im AKKU 1 wird ein Fehlercode übergeben, der den auslösenden Weckfehler näher spezifiziert:

1 Bearbeitung des OB 6
2 Bearbeitung des OB 13
3 Bearbeitung des OB 12
4 Bearbeitung des OB 11
5 Bearbeitung des OB 10

Ist der Weckfehler OB 33 nicht programmiert, erfolgt keine Reaktion auf den gemeldeten Weckfehler.

**OB 34** **Reaktionen auf Signal BAU (Batterieausfall)**

Das AG überprüft ständig den Zustand der Batterie in der Stromversorgung. Wenn ein Batterieausfall (BAU) eintritt, wird vor jedem Zyklus der OB 34 bearbeitet. Im OB 34 programmieren Sie die Reaktion bei Batterieausfall. Ist der OB 34 nicht programmiert, so erfolgt keine Reaktion.

**Hinweis!**

Bei einem Fehler der eingebauten Batterie muss die Hardware an die VIPA gesandt werden.

Weitere OBs, die Ihnen Betriebsfunktionen zur Verfügung stellen sind:

| | |
|---|---|
| OB 160 | Zeitschleife |
| OB 251 | PID-Regelalgorithmus. |

Eine Beschreibung finden Sie im Teil "Integrierte Bausteine".

# Bearbeiten von Bausteinen

In den vorangegangenen Abschnitten wurde bereits beschrieben, wie Bausteine eingesetzt werden können.

Bereits programmierte Bausteine können natürlich wieder verändert werden. Die einzelnen Änderungsmöglichkeiten werden nur kurz beschrieben. In der Bedienungsanleitung des verwendeten PGs bzw. in Ihrer Programmiersoftware werden die notwendigen Arbeitsschritte ausführlich erklärt.

## Programmänderungen

Programmänderungen können - unabhängig von der Bausteinart - in folgenden PG-Funktionen durchgeführt werden:

- EINGABE
- AUSGABE
- STATUS.

In diesen Funktionen können Sie folgende Änderungen vornehmen:

- Anweisungen löschen, einfügen oder überschreiben,
- Netzwerke einfügen oder löschen.

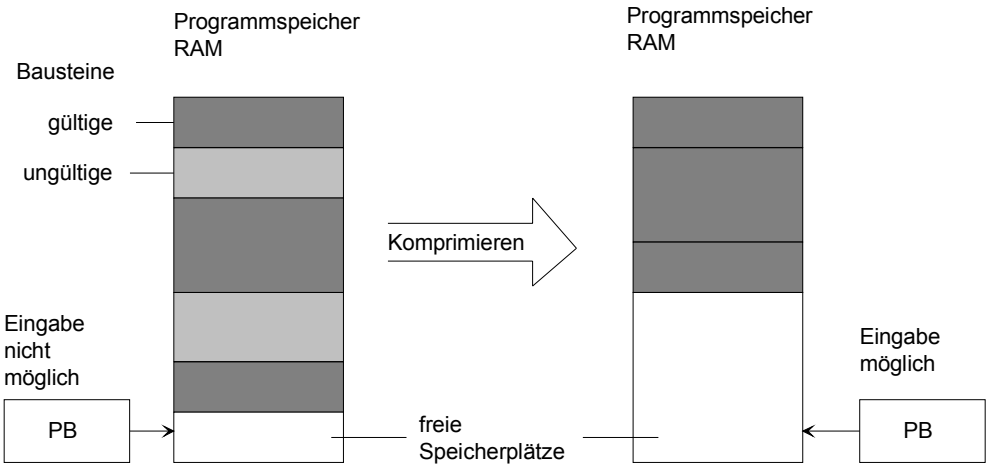
## Bausteinänderungen

Programmänderungen beziehen sich auf den Inhalt eines Bausteins. Sie können aber auch ganze Baustein löschen oder überschreiben. Dabei werden die Bausteine jedoch nicht im Programmspeicher gelöscht, sondern lediglich ungültig gemacht. Diese Speicherplätze können nicht neu beschrieben werden. Diese Tatsache kann dazu führen, dass neue Bausteine nicht mehr angenommen werden; es erfolgt über das PG die Fehlermeldung "Kein Speicherplatz".

Beseitigen Sie dies, indem Sie den AG-Speicher komprimieren.

## Programm-speicher komprimieren

Das folgende Bild zeigt, was bei der Operation KOMPRIMIEREN im Programmspeicher geschieht. Intern wird ein Baustein pro Zyklus verschoben.




Sie können den internen Programmspeicher komprimieren:

- entweder mit der PG-Funktion KOMPRIMIEREN

oder

- mit dem integrierten FB 238 (COMPR).

Wenn während des Schiebens eines Bausteins beim Komprimieren ein Netzausfall auftritt und das Baustein-Schieben nicht beendet werden kann, bleibt die CPU mit der Fehlermeldung NINEU im STOP-Zustand. Neben NINEU sind im USTACK die Bits BSTSCH, SCHTAE und SPABBR gesetzt.

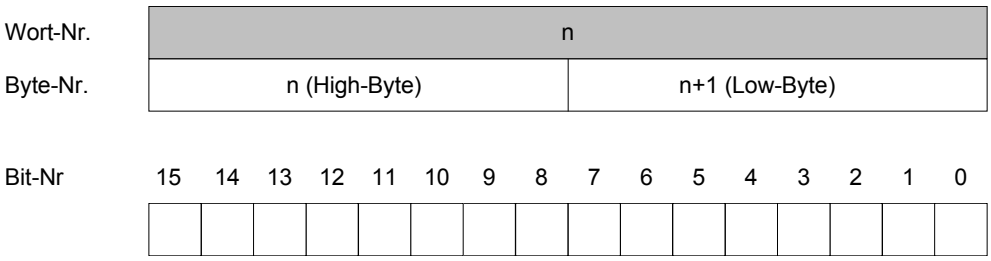
Abhilfe: Urlöschen.

# Zahlendarstellung

Die Programmiersprache gibt Ihnen die Möglichkeit, mit Zahlen in fünf verschiedenen Darstellungen zu arbeiten:

- Dezimalzahlen von -32768 bis +32767 (KF)
- Hexadezimalzahlen von 0000 bis FFFF (KH)
- BCD-codierte Zahlen (4 Tetraden) von 0000 bis 9999
- Bitmuster (KM)
- Konstante Byte (KY) von 0,0 bis 255,255.

Das Automatisierungsgerät stellt intern alle Zahlen als 16-stellige Dualzahlen oder als Bitmuster dar.
Negative Zahlen werden im Zweier-Komplement dargestellt.




In der folgenden Tabelle sehen Sie drei Beispiele für die Zahlendarstellung im AG:

| Eingegebener Wert | Darstellung im AG |
|---|---|
| KF -50 | 1111 1111 1100 1110 |
| KH A03F | 1010 0000 0011 1111 |
| KY 3,10 | 0000 0011 0000 1010 |

# Fehlerdiagnose im Programm

Für eine Störung des Programms kann es verschiedene Ursachen geben. Stellen Sie zuerst fest, ob der Fehler bei der CPU, dem Programm oder den Peripheriebaugruppen liegt.

| Fehlerbild | Fehleranalyse |
|---|---|
| CPU in STOP<br>rote LED leuchtet | Störung in der CPU<br>Unterbrechungsanalyse mit dem PG durchführen. |
| CPU in RUN<br>grüne LED leuchtet<br>fehlerhafter Betrieb | Programmfehler: Fehleradresse bestimmen<br>Störung der Peripherie: Störungsanalyse durchführen. |

**Hinweis!**

Zur groben Unterscheidung zwischen einem AG-Fehler und einem Programmfehler programmieren Sie nur ein "BE" im OB 1.

Ein fehlerfreie CPU 24x geht dann bei Neustart in RUN.

**Achtung!**

Das direkte Verändern des internen Programmspeichers durch die PG Funktion AUSGABE ADR ist mit Risiken verbunden.

Es können z.B. im RUN-Zustand der CPU Speicherbereiche überschrieben werden (z.B. BSTACK), die dann zum "CPU-Absturz" führen.

Vermeiden Sie derartige Risiken, indem Sie:

- nur den dokumentierten Systemdatenbereich verändern (siehe Teil 2 "USTACK- Ausgabe")
- den Systemdatenbereich nur durch das Steuerungsprogramm verändern!

**Programmfehler**

Störungen, deren Ursache in einem fehlerhaften Programm liegen.

| Fehlerbild | Fehlerbeseitigung |
|---|---|
| Alle Eingänge sind Null.<br>Ausgänge werden nicht gesetzt. | Programm überprüfen. |
| Ein Eingang ist Null, ein Ausgang wird nicht gesetzt. | Programm auf Belegung überprüfen (Doppelbelegung, Flankenbildung). |
| Zeit oder Zähler läuft nicht oder fehlerhaft. | |
| Neustart fehlerhaft. | Neustartbausteine OB 21 / 22 überprüfen oder einfügen. |
| Sporadische Fehlfunktion. | Programm mit STATUS überprüfen. |

# Teil 9 Operationen

**Überblick**

In diesem Kapitel wird der Befehlsvorrat beschrieben. Grundoperationen, Ergänzende Operationen und Systemoperationen werden anhand von Beispielen näher erläutert.

**Inhalt**


| Thema | Seite |
|---|---|
| **Teil 9 Operationen** | **9-1** |
| Einleitung | 9-2 |
| Grundoperationen | 9-3 |
| Ergänzende Operationen | 9-40 |
| Systemoperationen | 9-66 |
| Anzeigenbildung | 9-76 |
| Programmbeispiele | 9-79 |

# Einleitung

Die Programmiersprache unterscheidet drei Arten von Operationen:

- Die **Grundoperationen** umfassen Funktionen, die in Organisations-, Programm-, Schritt- und Funktionsbausteinen ausgeführt werden können. Bis auf die Addition (+F), die Subtraktion (-F) und die organisatorischen Operationen können sie in allen drei Darstellungsarten (AWL, FUP, KOP) ein- und ausgegeben werden.

- Die **ergänzenden Operationen** beinhalten komplexe Funktionen, wie z.B. Substitutionsanweisungen, Prüffunktionen, Schiebe- und Umwandlungsoperationen. Sie können nur in der Darstellungsart AWL ein- und ausgegeben werden.

- **Systemoperationen** greifen direkt auf das Betriebsystem zu. Nur ein erfahrener Programmierer sollte sie einsetzen. Ein- und Ausgeben können Sie die Systemoperationen nur in der Darstellungsart AWL.

# Grundoperationen

**Verknüpfungs-operationen** In der folgenden Tabelle sind die einzelnen Operationen aufgelistet; Beispiele finden Sie auf den nächsten Seiten.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| O | | | **Oder-Verknüpfung von UND-Funktionen**<br>Das VKE der nächsten UND-Verknüpfung wird mit dem bisherigen UND nach ODER verknüpft. |
| U( | | | **UND-Verknüpfung von Klammerausdrücken**<br>Das VKE des Klammerausdrucks wird mit dem vorherigen VKE nach UND verknüpft. |
| O( | | | **ODER-Verknüpfung von Klammerausdrücken**<br>Das VKE des Klammerausdrucks wird mit dem vorherigen VKE nach ODER verknüpft. |
| ) | | | **Klammer zu**<br>Mit dieser Operation wird ein Klammerausdruck abgeschl.. |
| U | **x** | **x** | **UND-Verknüpfung, Abfrage auf Signalzustand "1"**<br>Das Abfrageergebnis ist "1", wenn der zugehörige Operand den Signalzustand "1" führt. Anderenfalls ist auch das Abfrageergebnis "0". Dieses Ergebnis wird mit dem VKE im Prozessor nach UND verknüpft. *1 |
| O | **x** | **x** | **ODER-Verknüpfung, Abfrage auf Signalzustand "1"**<br>Das Abfrageergebnis ist "1", wenn der zugehörige Operand den Signalzustand "1" führt. Anderenfalls ist auch das Abfrageergebnis "0". Dieses Ergebnis wird mit dem VKE im Prozessor nach ODER verknüpft. *1 |
| UN | **x** | **x** | **UND-Verknüpfung, Abfrage auf Signalzustand "0"**<br>Das Abfrageergebnis ist "1", wenn der zugehörige Operand den Signalzustand "0" führt. Anderenfalls ist das Abfrageergebnis "0". Dieses Ergebnis wird mit dem VKE im Prozessor nach UND verknüpft. *1 |
| ON | **x** | **x** | **ODER-Verknüpfung, Abfrage auf Signalzustand "0"**<br>Das Abfrageergebnis ist "1", wenn der zugehörige Operand den Signalzustand "0" führt. Anderenfalls ist das Abfrageergebnis "0". Dieses Ergebnis wird mit dem VKE im Prozessor nach ODER verknüpft. *1 |

| Kennzeichen | Parameter CPU 241...244 |
|---|---|
| E | 0.0...127.7 |
| A | 0.0...127.7 |
| M | 0.0 255.7 |
| S | 0.0...1023.7 |
| T | 0...127 |
| Z | 0...127 |

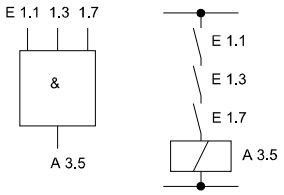
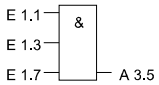
* 1 Folgt die Abfrage unmittelbar auf eine VKE begrenzenden Operation (Erstabfrage), so wird das Abfrageergebnis als neues VKE übernommen.

**Verknüpfungs-operationen**

## UND-Verknüpfung

Mit dieser Operation wird abgefragt, ob verschiedene Bedingungen gleichzeitig erfüllt sind.

*UND-Verknüpfung*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|  | U E 1.1<br>U E 1.3<br>U E 1.7<br>= A 3.5 | |  |





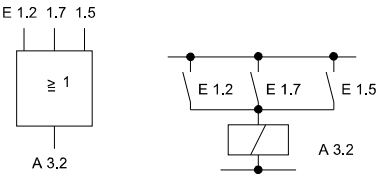
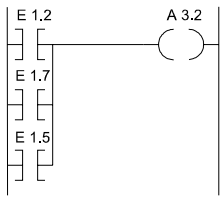
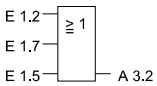


Am Ausgang A 3.5 erscheint Signalzustand "1", wenn alle Eingänge gleichzeitig den Signalzustand "1" aufweisen.

Am Ausgang A 3.5 erscheint Signalzustand "0", wenn mindestens einer der Eingänge den Signalzustand "1" aufweist.

Anzahl der Abfragen und die Reihenfolge der Programmierung ist beliebig.

*ODER-Verknüpfung*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|  | O E 1.2<br>O E 1.7<br>O E 1.5<br>= A 3.2 |  |  |





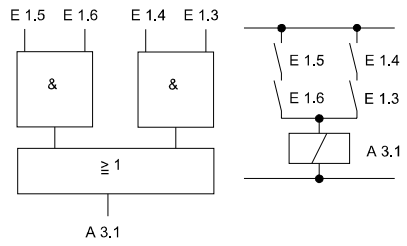
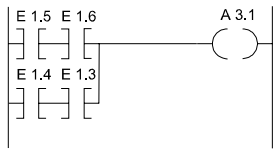
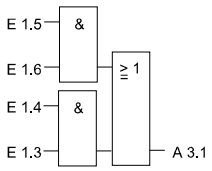


Am Ausgang A 3.2 erscheint Signalzustand "1", wenn mindestens einer der Eingänge den Signalzustand "1" aufweist.

Am Ausgang A 3.2 erscheint Signalzustand "0", wenn alle Eingänge gleichzeitig den Signalzustand "0" aufweisen.

Anzahl der Abfragen und die Reihenfolge der Programmierung ist beliebig.

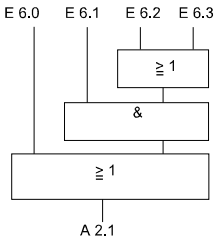
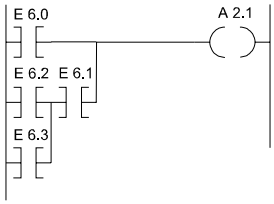
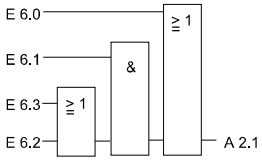
*UND-vor-ODER-Verknüpfung*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|   | U E 1.5<br>U E 1.6<br>O<br>U E 1.4<br>U E 1.3<br>= A 3.1 |   |   |

Am Ausgang A 3.1 erscheint Signalzustand "1", wenn mindestens eine UND-Verknüpfung erfüllt ist.

Am Ausgang A 3.1 erscheint Signalzustand "0", wenn keine UND-Verknüpfung erfüllt ist.

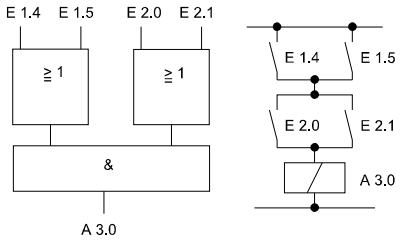
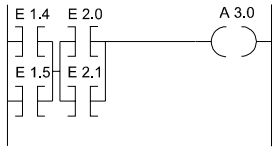
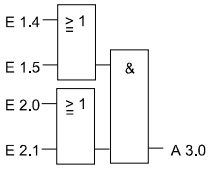
*ODER-vor-UND-Verknüpfung* *1. Beispiel*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|   | U E 6.0<br>O<br>U E 6.1<br>U(<br>O E 6.2<br>O E 6.3<br>)<br>= A 2.1 |   |   |

Am Ausgang A 2.1 erscheint Signalzustand "1", wenn Eingang E 6.0 oder Eingang E 6.1 und einer der Eingänge E 6.2 bzw. E 6.3 Signal "1" führen.

Am Ausgang A 2.1 erscheint Signalzustand "0", wenn Eingang E 6.0 Signal "0" führt und die UND-Verknüpfung nicht erfüllt ist.

*ODER-vor-UND-Verknüpfung* *2.Beispiel*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|  | U(<br>O E 1.4<br>O E 1.5<br>)<br>U(<br>O E 2.0<br>O E 2.1<br>)<br>= A 3.0 |  |  |



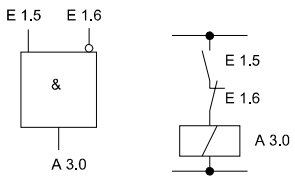
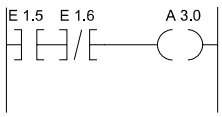
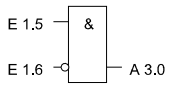




Am Ausgang A 3.0 erscheint Signalzustand "1", wenn beide ODER-Verknüpfungen erfüllt sind.

Am Ausgang A 3.0 erscheint Signalzustand "0", wenn mindestens eine ODER-Verknüpfung nicht erfüllt ist.

*Abfrage auf Signalzustand "0"*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|  | U E 1.5<br>UN E 1.6<br>= A 3.0 |  |  |







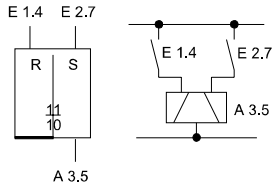
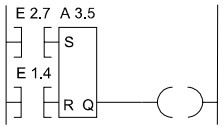
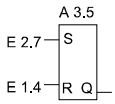
Am Ausgang A 3.0 erscheint Signalzustand "1" nur dann, wenn Eingang E 1.5 den Signalzustand "1" (Schließer betätigt) und der Eingang E 1.6 den Signalzustand "0" (Öffner nicht betätigt) führt.

**Speicher-operationen**

Mit Speicheroperationen wird das im Steuerwerk gebildete Verknüpfungsergebnis als Signalzustand des angesprochenen Operanden gespeichert. Das Speichern kann dynamisch (Zuweisen) oder statisch (Setzen und Rücksetzen) erfolgen. Die folgend Tabelle gibt Ihnen eine Überblick über die einzelnen Operationen; Beispiele finden Sie auf den folgenden Seiten.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| S | **x** | **x** | **Setzen**<br>Bei der ersten Programmbearbeitung auf VKE = "1" wird dem angesprochenen Operanden der Signalzustand "1" zugewiesen. Änderungen des VKE ändern diesen Zustand nicht mehr. |
| R | **x** | **x** | **Rücksetzen**<br>Bei der ersten Programmbearbeitung mit VKE = "1" wird dem angesprochenen Operanden der Signalzustand "0" zugewiesen. Ein Wechsel beim VKE ändert diesen Zustand nicht. |
| = | **x** | **x** | **Zuweisen**<br>Bei jeder Programmbearbeitung wird dem angesprochenen Operanden das aktuelle VKE zugewiesen. |
| Kennzeichen | | Parameter CPU 241...244 | |
| | E | 0.0...127.7 | |
| | A | 0.0...127.7 | |
| | M | 0.0...255.7 | |
| | S | 0.0...1023.7 | |

*RS-Speicherglied für speichernde Signalausgabe*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|   | U E 2.7<br>S A 3.5<br>U E 1.4<br>R A 3.5 |   |   |

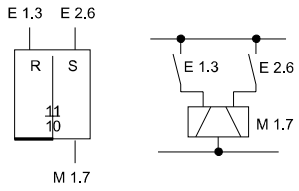
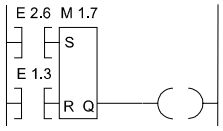
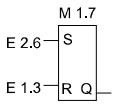
Signalzustand "1" am Eingang E 2.7 bewirkt das Setzen des Speichergliedes (Signalzustand "1" am Ausgang A 3.5).

Wechselt der Signalzustand am Eingang E 2.7 nach "0", so bleibt dieser Zustand erhalten, d.h., das Signal wird gespeichert.

Signalzustand "1" am Eingang E 1.4 bewirkt das Rücksetzen des Speichergliedes (Signalzustand "0" am Ausgang A 3.5).

Wechselt der Signalzustand am Eingang E 1.4 nach "0", so bleibt dieser Zustand erhalten. Bei gleichzeitigem Anliegen des Setzsignals (Eingang E 2.7) und des Rücksetzsignals (Eingang E 1.4) ist die zuletzt programmierte Abfrage (hier U E 1.4) während der Bearbeitung des übrigen Programms wirksam. Das Rücksetzen ist vorrangig.

*RS-Speicherglied mit Merkern*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|   | U E 2.6<br>S M 1.7<br>U E 1.3<br>R M 1.7 |   |   |

Signalzustand "1" am Eingang E 2.6 bewirkt das Setzen des Speichergliedes.

Wechselt der Signalzustand am Eingang E 2.6 nach "0", so bleibt dieser Zustand erhalten, d.h. das Signal wird gespeichert.

Signalzustand "1" am Eingang E 1.3 bewirkt das Rücksetzen des Speichergliedes.

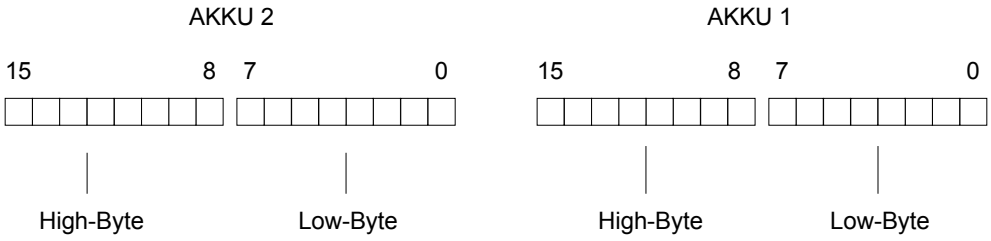
Wechselt der Signalzustand am Eingang E 1.3 nach "0", so bleibt dieser Zustand erhalten. Bei gleichzeitigem Anliegen des Setzsignals (Eingang E 2.6) und des Rücksetzsignals (Eingang E 1.3) ist die zuletzt programmierte Abfrage (hier U E 1.3) während der Bearbeitung des übrigen Programms wirksam. Das Rücksetzen ist vorrangig.

**Laden und Transferieren**

Mit Lade- und Transferoperationen können Sie

- Informationen zwischen den verschiedenen Operandenbereichen austauschen,
- Zeit- und Zählwerte für die Weiterverarbeitung vorbereiten,
- konstante Werte für die Programmbearbeitung laden.

Der Informationsfluss erfolgt indirekt über Akkumulatoren (AKKU 1 und AKKU 2). Die Akkumulatoren sind besondere Register in der CPU, die als Zwischenspeicher dienen. Bei der CPU 24x haben sie eine Länge von jeweils 16 Bit. Die Struktur der Akkumulatoren wird im folgenden Bild verdeutlicht.




Zulässige Operanden können byte- oder wortweise geladen und transferiert werden. Beim byteweisen Austausch werden die Informationen rechtsbündig, also im Low-Byte, abgelegt.

Die restlichen Bits werden auf Null gesetzt.

Die Informationen in den beiden Akkumulatoren können mit verschiedenen Operationen bearbeitet werden.

Lade- und Transferoperationen werden unabhängig von den Anzeigen durchgeführt; die Anzeigen werden bei der Ausführung der Operationen nicht beeinflusst.

Sie können nur in Zusammenhang mit Zeit oder Zähloperationen graphisch programmiert werden; sonst ist nur eine Darstellung in AWL möglich.

**Laden und Transferieren**

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| L | **x** | **x** | **Laden**<br>Die Operandeninhalte werden unabhängig vom VKE in den AKKU 1 kopiert.<br>Das VKE wird nicht beeinflusst. |
| T | **x** ↑ | **x** ↑ | **Transferieren**<br>Der Inhalt von AKKU 1 wird unabhängig vom VKE einem Operanden zugewiesen.<br>Das VKE wird nicht beeinflusst. |
| Kennzeichen | | Parameter CPU 24x | |
| | EB | 0...127 | |
| | EW | 0...126 | |
| | AB | 0...127 | |
| | AW | 0...126 | |
| | MB | 0...255 | |
| | MW | 0...254 | |
| | SY | 0...1023 | |
| | SW | 0...1022 | |
| | DR | 0...255 | |
| | DL | 0...255 | |
| | DW | 0...254 | |
| | T *1 | 0...127 | |
| | Z *1 | 0...127 | |
| | PB | 0...127 | |
| | PY *2 | 128...255 | |
| | PW | 0...126<br>128...254 | |
| | KM*1 | beliebiges Bitmuster (16 Bit) | |
| | KH*1 | 0...FFFF | |
| | KF*1 | -32768...+32767 | |
| | KY*1 | 0...255 je Byte | |
| | KB*1 | 0...255 | |
| | KC*1 | 2 beliebige alphanumerische Zeichen | |
| | KT*1 | 0.0...999.3 | |
| | KZ*1 | 0...999 | |
| LC | **x** ↑ | **x** ↑ | **Lade codiert**<br>Duale Zeit- und Zählwerte werden unabhängig vom VKE BCD-codiert in den AKKU 1 geladen. |
| Kennzeichen | | Parameter | |
| | T | 0...127 | |
| | Z | 0...127 | |

*1 nicht bei "Transferieren"

*2 PY bei S5-DOS-PG von Siemens

**Laden und Transferieren**

## Laden

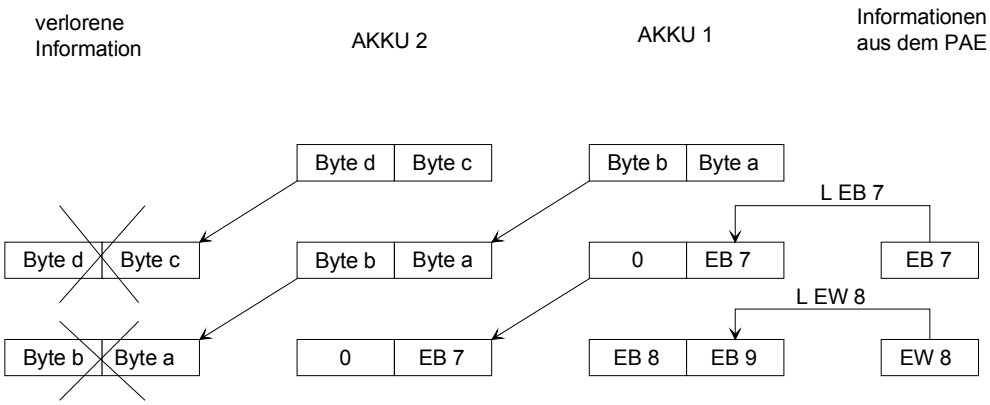
Beim Laden wird die Information aus dem jeweiligen Speicherbereich - z.B. aus dem PAE - in den AKKU 1 kopiert.

Der vorherige Inhalt von AKKU 1 wird in den AKKU 2 geschoben.

Der ursprüngliche Inhalt von AKKU 2 geht verloren.

**Beispiel**: Nacheinander werden zwei Bytes (EB 7 und EB 8) aus dem PAE in den Akkumulator geladen.

Das PAE wird dabei nicht verändert.




## Transferieren:

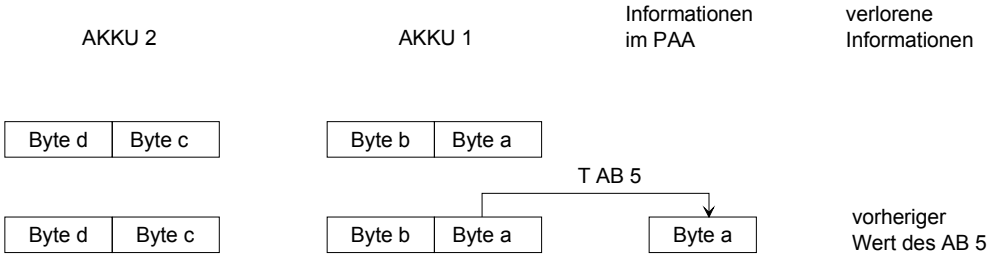
Beim Transferieren wird die Information aus dem AKKU1 in den angesprochenen Speicherbereich -z.B. ins PAA - kopiert.

Der Inhalt des AKKU 1 bleibt dabei unverändert.

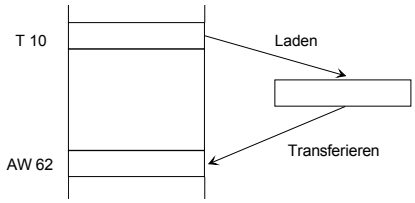
Beim Transferieren in den Digital-Ausgabebereich wird das zugehörige Byte oder Wort im PAA automatisch aktualisiert.

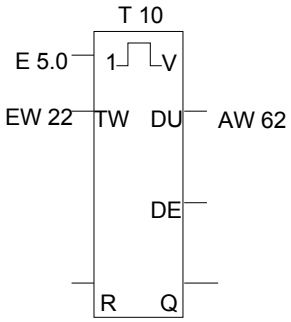
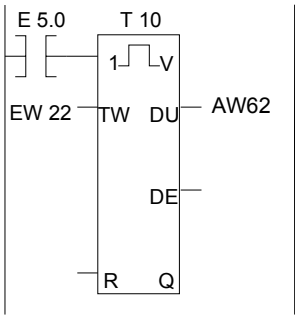
Beispiel:

Das Bild zeigt, wie das Byte a - das Low-Byte in AKKU 1 - zum AB 5 transferiert wird.



**Laden und Transferieren** **Laden und Transferieren eines Zeitwertes**

| Beispiel | Darstellung |
|---|---|
| Bei der graphischen Eingabe wurde der Ausgang DU des Zeitwertes mit AW 62 belegt. Das Programmiergerät hinterlegt daraufhin selbsttätig den entsprechenden Lade- und Transferbefehl im Steuerungsprogramm. So wird der Inhalt der mit T 10 adressierten Speicherzelle in den AKKU 1 geladen.<br>Anschließend wird der Akkumulatorinhalt in das mit AW 62 adressierte Prozessabbild transferiert.<br>Am AW 62 sieht man in diesem Beispiel die Zeit 10 dualcodiert mitlaufen.<br>Die Ausgänge DU und DE sind digitale Ausgänge. Am Ausgang DU steht der Zeitwert dual-codiert, am Ausgang DE BCD-codiert mit Zeitraster an. |  |

| AWL | FUP | KOP |
|---|---|---|
| U E 5.0<br>L EW 22<br>SI T 10<br>NOP 0<br>L T 10<br>T AW 62<br>NOP 0<br>NOP 0 |  |  |

**Zeitoperationen** Mit den Zeitoperationen werden zeitliche Abläufe durch das Programm realisiert und überwacht. In der folgenden Tabelle sind die einzelnen Zeitoperationen aufgelistet; Beispiele finden Sie auf den nächsten Seiten.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| SI | **x** | **x** | **Starten einer Zeit als Impuls**<br>Die Zeit wird bei steigender Flanke des VKE gestartet.<br>Bei VKE "0" wird die Zeit auf "0" gesetzt.<br>Abfragen liefern Signalzustand "1", solange die Zeit läuft. |
| SV | **x** | **x** | **Starten einer Zeit als verlängerter Impuls**<br>Die Zeit wird bei steigender Flanke des VKE gestartet.<br>Bei VKE "0" bleibt die Zeit unbeeinflusst.<br>Abfragen liefern Signalzustand "1", solange die Zeit läuft. |
| SE | **x** | **x** | **Starten einer Zeit als Einschaltverzögerung**<br>Die Zeit wird bei steigender Flanke des VKE gestartet.<br>Bei VKE "0" wird die Zeit auf "0" gesetzt.<br>Abfragen liefern Signalzustand "1", wenn die Zeit abgelaufen ist und das VKE am Eingang noch ansteht. |
| SS | **x** | **x** | **Starten einer Zeit als speichernde Einschaltverzögerung**<br>Die Zeit wird bei steigender Flanke des VKE gestartet.<br>Bei VKE "0" bleibt die Zeit unbeeinflusst.<br>Abfragen liefern Signalzustand "1", wenn die Zeit abgelaufen ist.<br>Der Signalzustand wird "0", wenn die Zeit mit der Operation "R" zurückgesetzt wurde. |
| SA | **x** | **x** | **Starten einer Zeit als Ausschaltverzögerung**<br>Die Zeit wird bei fallender Flanke des VKE gestartet.<br>Bei VKE "1" wird die Zeit auf den Anfangswert gesetzt.<br>Abfragen liefert Signalzustand "1", solange das VKE am Eingang "1" ist oder die Zeit läuft. |
| R | **x** | **x** | **Rücksetzen einer Zeit**<br>Die Zeit wird auf den Anfangswert zurückgesetzt, solange das VKE "1" ist.<br>Bei VKE "0" bleibt die Zeit unbeeinflusst.<br>Abfragen liefern den Signalzustand "0", solange die Zeit zurückgesetzt wird oder noch nicht "gestartet" wurde. |
| Kennzeichen<br>T | | Parameter<br>0...127 | |

**Zeitoperationen**

**Laden eines Zeitwertes**

Die Operationen rufen die internen Zeitgeber auf.

Beim Starten einer Zeitoperation wird das im AKKU 1 stehende Wort als Zeitwert übernommen. Deshalb müssen zuerst Zeitwerte im Akkumulator festgelegt werden.

Ein Zeitgeber kann geladen werden mit einem:

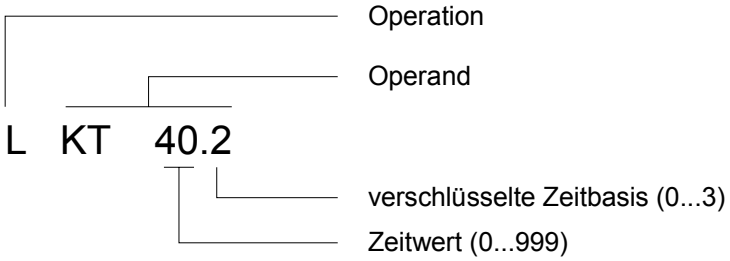
KTkonstanten Zeitwert

oder

DW Datenwort
EW Eingangswort
AW Ausgangswort
MW Merkerwort
SW Schmiermerkerwort

**Ein konstanter Zeitwert wird geladen:**

Das folgende Beispiel zeigt, wie Sie eine Zeitwert von 40s laden können.




Schlüssel für Zeitbasis:

| Basis | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| Faktor | 0,01s | 0,1s | 1s | 10s |

**Zeitoperationen**

**Beispiel:**
KT 40.2 entspricht 40 x 1s

Toleranzen:

Die Zeitwerte besitzen eine Unschärfe in Höhe der Zeitbasis.

| Beispiele | Operand | Zeitintervall | | |
|---|---|---|---|---|
| Einstell- | KT 400.1 | 400 x 0,1s | - 0,1s | 39,9s...40s |
| möglichkeiten | KT 40.2 | 40 x 1s | - 1s | 39s...40s |
| für die Zeit 40s | KT 4.3 | 4 x 10s | - 10s | 30s...40s |

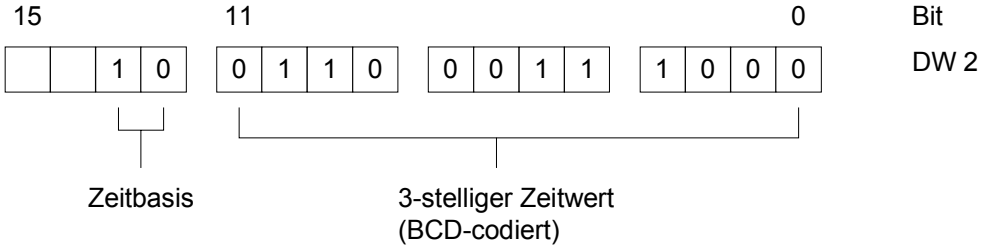
**Hinweis!**

Verwenden Sie immer die kleinstmögliche Zeitbasis

Ein Zeitwert wird als Eingangs-, Ausgangs-, Merker- oder Datenwort geladen:

Lade-Anweisung:L DW 2

Im Datenwort 2 ist der Zeitwert 638s BCD-codiert hinterlegt.
Die Bits 14 und 15 sind für den Zeitwert ohne Bedeutung.




Schlüssel für Zeitbasis:

| Basis | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| Faktor | 0,01s | 0,1s | 1s | 10s |

Das Datenwort 2 kann auch durch das Steuerungsprogramm beschrieben werden.

Beispiel: Der Wert 270 x 100ms soll im Datenwort 2 des DB 3 abgelegt werden.

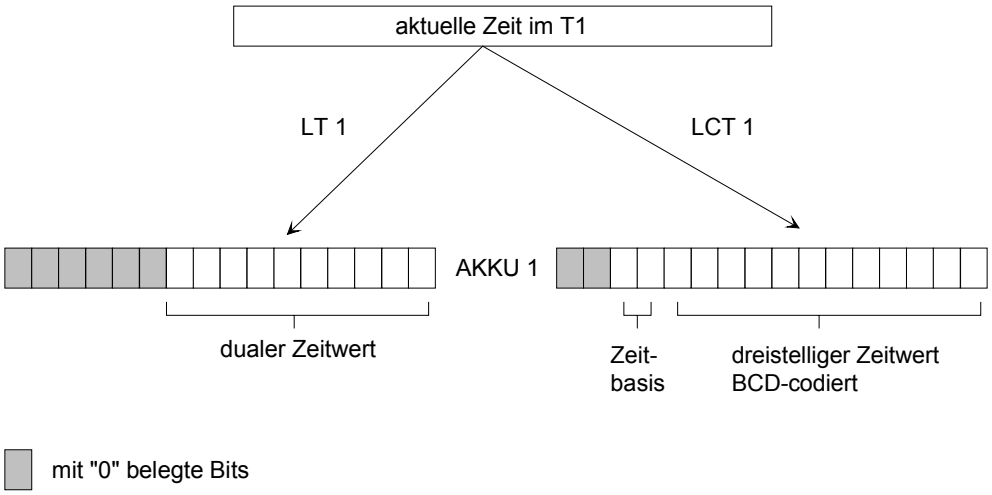
```
A  DB 3
L  KT 270.1
T  DW 2
```

**Zeitoperationen**

Ausgabe der aktuellen Zeit:

Die aktuelle Zeit kann durch eine Ladeoperation in den AKKU 1 geladen und von hier aus weiterverarbeitet werden.

Für die Ausgabe über eine Ziffernanzeige eignet sich die Operation "Lade codiert".

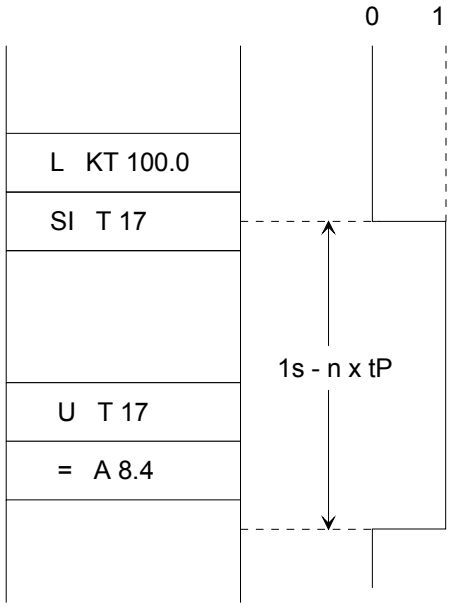


**Zeitoperationen**

**Starten einer Zeit:**

Die Zeiten laufen im AG asynchron zur Programmbearbeitung ab. Die eingestellte Zeit kann während einer Programmbearbeitung abgelaufen sein. Die Auswertung erfolgt durch die nächste Zeitabfrage. Dazwischen liegt im ungünstigsten Fall eine ganze Programmbearbeitung. Zeitglieder sollten deshalb nicht durch sich selbst angestoßen werden.

**Beispiel:**

| Schematische Darstellung | Erläuterung |
|---|---|
|  <br>n: Anzahl der Programmbearbeitungen<br><br>tP: Programmbearbeitungszeit | Das Bild zeigt die "n+1"-te Bearbeitung seit dem Start der Zeit T17 (KT 100.0 entspricht 1s). Obwohl die Zeit kurz nach der Anweisung "= A 8.4" abgelaufen ist, bleibt der Ausgang 8.4 gesetzt. Erst bei der nächsten Programmbearbeitung wird diese Änderung berücksichtigt<br><br>. |

Außer der Operation "Rücksetzen einer Zeit" werden alle Zeitoperationen nur bei einem Flankenwechsel - das VKE wechselt zwischen "0" und "1" gestartet.

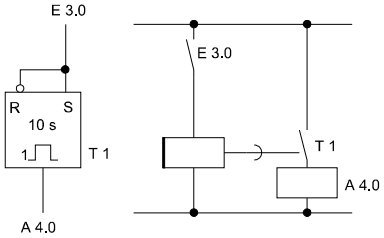
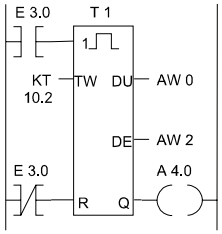
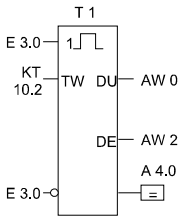
Nach dem Start wird der geladenen Zeitwert im Takt der Zeitbasis jeweils um eine Einheit erniedrigt, bis er den Wert Null erreicht hat.

Wird die Flanke gewechselt, während die Zeit noch läuft, so wird die Zeit wieder auf den Anfangswert gesetzt und gestartet.

Der Signalzustand eines Zeitgliedes kann mit Verknüpfungsoperationen abgefragt werden.

## Zeitoperationen Impuls

*Impuls*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|   | U E 3.0<br>L KT 10.2<br>SI T 1<br>UN E 3.0<br>R T 1<br>L T 1<br>T AW 0<br>LC T 1<br>T AW 2<br>U T 1<br>= A 4.0 |   |   |

Bei Signalzustandswechsel von "0" nach "1" am Eingang E 3.0 wird das Zeitglied gestartet. Bei wiederholter Bearbeitung mit Verknüpfungsergebnis "1" bleibt das Zeitglied unbeeinflusst.

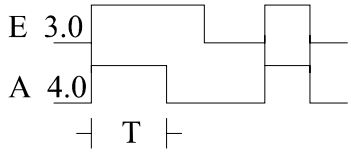
Bei Signalzustand "0" am Eingang E 3.0 wird das Zeitglied auf Null gesetzt (gelöscht).

Die Abfragen U T bzw. O T liefern Signalzustand "1", solange die Zeit läuft.

KT 10.2

Das Zeitglied wird mit dem angegebenen Wert (10) geladen.

Die Zahl rechts vom Punkt gibt das Zeitraster an:

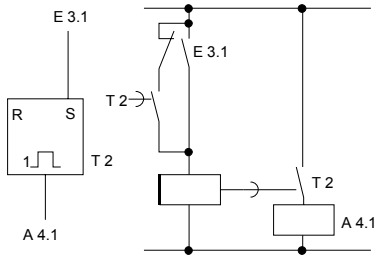
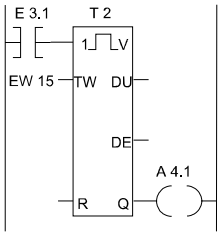
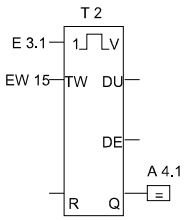



0 = 001 s 2 = 1 s

1 = 0.1 s 3 = 10 s

DU und DE sind digitale Ausgänge der Zeitzelle. Am Ausgang DU steht der Zeitwert dualcodiert, am Ausgang DE BCD-codiert mit Zeitraster an.

*Verlängerter Impuls*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|  | U E 3.1<br>L EW 15<br>SV T 2<br>U T 2<br>= A 4.1 |  |  |

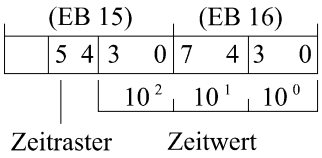






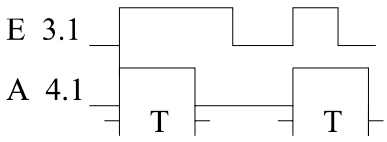
Bei Verknüpfungsergebnis "1" und erstmaliger Bearbeitung wird das Zeitglied gestartet.

Bei Verknüpfungsergebnis "0" bleibt das Zeitglied unbeeinflusst.
Die Abfragen U T bzw. O T liefern Signalzustand "1",

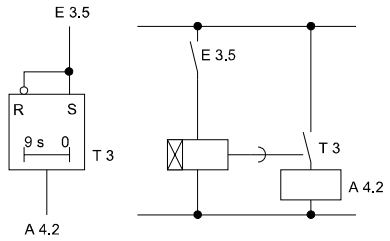
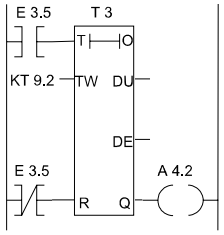
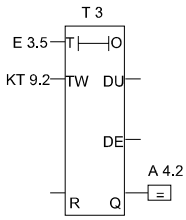



solange die Zeit läuft.

EW 15:
Setzen des Zeitwerts mit dem im BCD-Code vorliegenden Wert der Operanden E, A, M oder D (im Beispiel Eingangswort 15).



*Einschaltverzögerung*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|  | U E 3.5<br>L KT 9.2<br>SE T 3<br>UN E 3.5<br>R T 3<br>U T 3<br>= A 4.2 |  |  |

Bei Verknüpfungsergebnis "1" und erstmaliger Bearbeitung wird das Zeitglied gestartet. Bei wiederholter Bearbeitung mit Verknüpfungsergebnis "1" bleibt das Zeitglied unbeeinflusst.

Bei Signalzustand "0" am Eingang E 3.5 wird das Zeitglied auf Null gesetzt (gelöscht).

Die Abfragen U T bzw. O T liefern Signalzustand "1", wenn die Zeit abgelaufen ist und der Signalzustand "1" am Eingang E 3.5 noch ansteht.

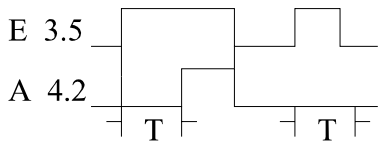
KT 9.2 :

Das Zeitglied wird mit dem angegeben Wert (9) geladen.

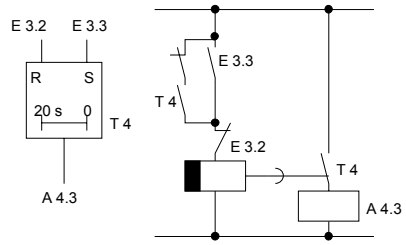
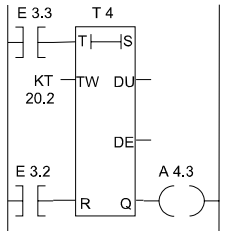
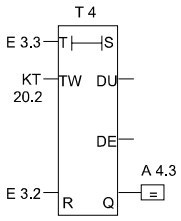
Die Zahl rechts vom Punkt gibt das Zeitraster an.

0 = 0.01 s 2 = 1 s

1 = 0.1 s 3 = 10 s

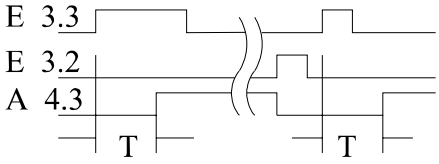


*Speichernde Einschaltverzögerung*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|  | U E 3.3<br>L KT20.2<br>SS T 4<br>U E 3.2<br>R T 4<br>U T 4<br>= A 4.3 |  |  |

Bei Verknüpfungsergebnis "1" und erstmaliger Bearbeitung wird das Zeitglied bearbeitet.

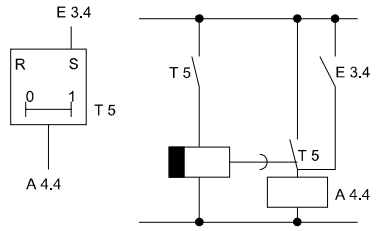
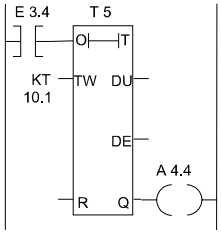
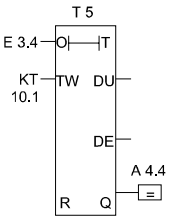
Bei Verknüpfungsergebnis "0" bleibt das Zeitglied unbeeinflusst.




Die Abfragen U T bzw. O T liefern Signalzustand "1", wenn die Zeit abgelaufen ist.

Der Signalzustand wird erst dann "0", wenn das Zeitglied mit der Funktion RT zurückgesetzt worden ist.

*Ausschaltverzögerung*

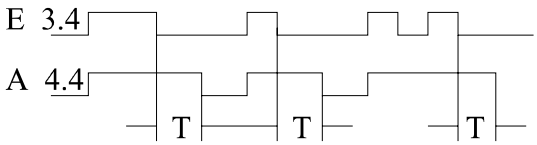
| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|  | U E 3.4<br>L KT 10.1<br>SA T 5<br>U T 5<br>= A 4.4 |  |  |







Wenn das Verknüpfungsergebnis am Starteingang von "1" nach "0" wechselt, wird die Zeit gestartet. Sie läuft mit der programmierten Zeitdauer ab.




Bei Verknüpfungsergebnis "1" wird das Zeitglied auf Null gesetzt (gelöscht).

Die Abfragen U T bzw. O T liefern Signalzustand "1", wenn die Zeit läuft oder das Verknüpfungsergebnis am Eingang "1" ist.

**Hinweis!**

Zeitwerte besitzen eine Unschärfe in Höhe der Zeitbasis.

**Zähloperationen**

Mit den Zähloperationen werden Zählaufgaben direkt von der CPU ausgeführt. Es kann vorwärts und rückwärts gezählt werden. Der Zählbereich liegt zwischen 0 und 999 (drei Dekaden). Die folgende Tabelle gibt eine Übersicht der Zähloperationen. Anschließend sind verschiedene Beispiele aufgeführt.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| S | **x** | **x** | **Setzen eines Zählers**<br>Der Zähler wird bei steigender Flanke des VKE gesetzt. |
| R | **x** | **x** | Rücksetzen eines Zählers<br>Der Zähler wird auf Null gesetzt, solange das VKE "1" ist. |
| ZV | **x** | **x** | **Vorwärtszählen eines Zählers**<br>Der Zählwert wird bei steigender Flanke um 1 erhöht. Bei VKE "0" bleibt der Zählwert unbeeinflusst |
| ZR | **x** | **x** | **Rückwärtszählen eines Zählers**<br>Der Zählwert wird bei steigender Flanke des VKE um 1 erniedrigt.<br>Bei VKE "0" bleibt der Zählwert unbeeinflusst |
| Kennzeichen | Z | Parameter | 0...127 |

**Laden eines Zählwertes**

Die Zähloperationen rufen die internen Zähler auf.

Beim Setzen eines Zählers wird das im AKKU 1 stehende Wort als Zählwert übernommen. Deshalb müssen zuerst Zählwerte im Akkumulator abgelegt werden.

Ein Zähler kann geladen werden mit einem:

KZkonstanten Zählwert

oder

DW Datenwort

EW Eingangswort

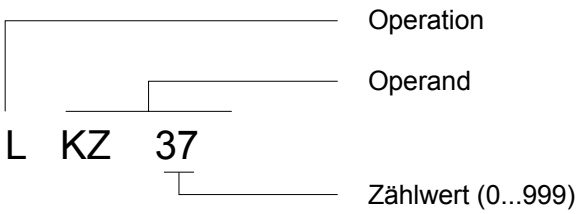
AW Ausgangswort

MW Merkerwort

SW Schmiermerkerwort.

**Zähloperationen**

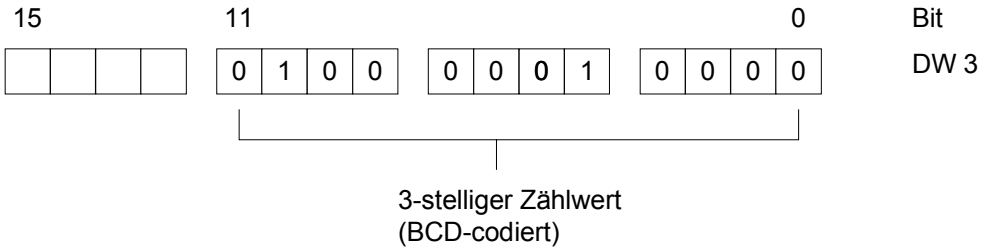
Ein konstanter Zählwert wird geladen:
Das folgende Beispiel zeigt, wie der Zählwert 37 geladen wird




Ein Zählwert wird als Eingangs-, Ausgangs-, Merker- oder Datenwort geladen:

Lade-Anweisung:L DW 3

Im Datenwort 3 ist der Zählwert 410 BCD-codiert hinterlegt.
Die Bits 12 bis 15 sind für den Zählwert ohne Bedeutung.




**Zähler abfragen:**

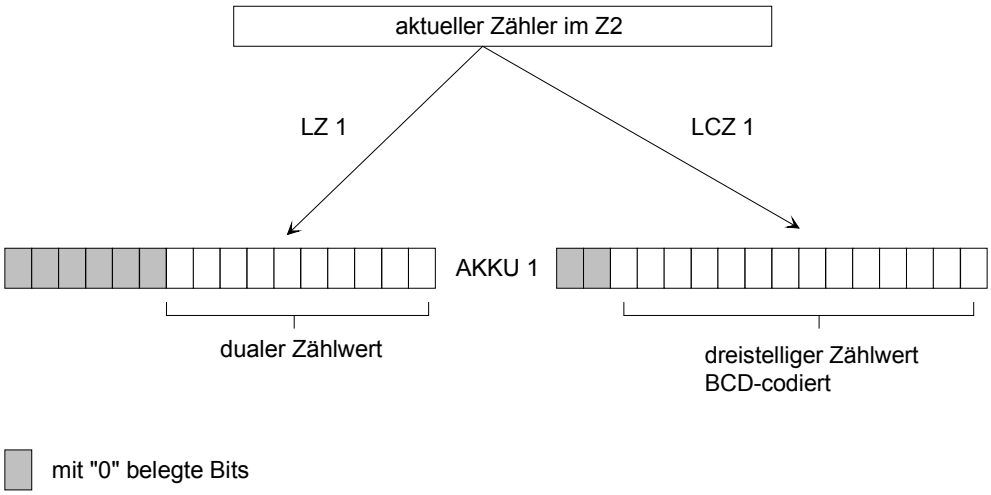
Der Zählerstand kann durch Verknüpfungsoperationen (z.B. U Zx) abgefragt werden. Solange der Zählwert von Null verschieden ist, erhält man als Abfrageergebnis den Signalzustand "1".

**Zähloperationen**

Ausgabe des aktuellen Zählerstandes

Der aktuelle Zählerstand kann durch eine Ladeoperation in den AKKU 1 geladen und von dort aus weiterverarbeitet werden.

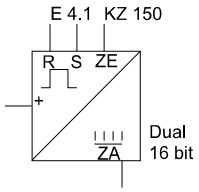
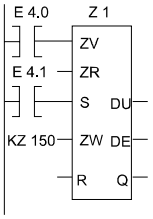
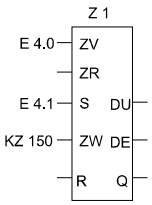
Für die Ausgabe über eine Ziffernanzeige eignet sich die Operation "Lade codiert".




mit "0" belegte Bits

**Zähloperationen** **Setzen eines Zählers "S" und Rückwärtszählen "ZR"**

*Zähler setzen*

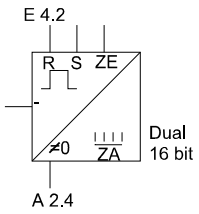
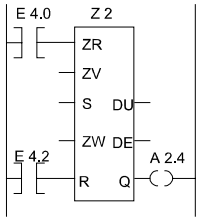
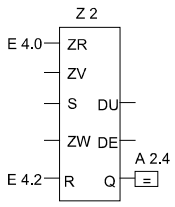
| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|  | U E 4.0<br>ZV Z 1<br>U E 4.1<br>L KZ 150<br>S Z 1 |  |  |

Wenn das Verknüpfungsergebnis am Starteingang (E 4.1) von "0" nach "1" wechselt, wird der Zähler mit dem angegebenen Wert (150) geladen.

Der für die Flankenauswertung des Setzeingangs erforderliche Merker ist im Zählwort mitgeführt.

DU und DE sind digitale Ausgänge der Zählerzelle. Am Ausgang DU steht der Zählwert dualcodiert, am Ausgang DE BCD-codiert an.

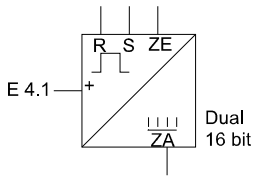
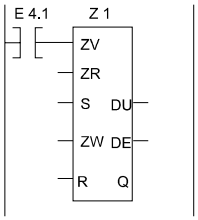
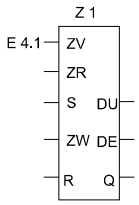
*Zähler rücksetzen*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|  | U E 4.0<br>ZR Z 2<br>U E 4.2<br>R Z 2<br>U Z 2<br>= A 2.4 |  |  |

Bei Verknüpfungsergebnis "1" (E 4.2) wird der Zähler auf Null gesetzt (rückgesetzt).

Bei Verknüpfungsergebnis "0" bleibt der Zähler unbeeinflusst.

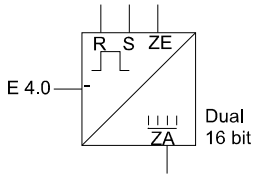
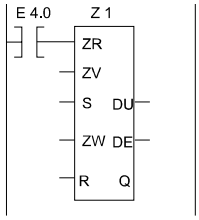
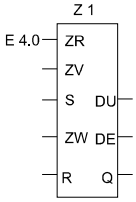
*Vorwärts zählen*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|  | U E 4.1<br>ZV Z 1 |  |  |

Der Wert des adressierten Zählers wird um 1 erhöht, maximal bis zum Zählwert 999. Die Funktion ZV wird nur bei einer positiven Flanke (von "0" nach "1") der vor ZV programmierten Verknüpfung ausgeführt. Die für die Flankenauswertung der Zähleingänge erforderlichen Merker sind im Zählwort mitgeführt.

Durch die zwei getrennten Flankenmerker für ZV und ZR kann ein Zähler mit zwei verschiedenen Eingängen als Vorwärts-/Rückwärtszähler verwendet werden.

*Rückwärts zählen*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|  | U E 4.0<br>ZR Z 1 |  |  |

Der Wert des adressierten Zählers wird um 1 erniedrigt, minimal bis zum Zählwert 0. Die Funktion wird nur bei einer positiven Flanke (von "0" nach "1") der vor ZR programmierten Verknüpfung ausgeführt. Die für die Flankenauswertung der Zähleingänge erforderlichen Merker sind im Zählwort mit geführt.

Durch die zwei getrennten Flankenmerker für ZV und ZR kann ein Zähler mit zwei verschiedenen Eingängen als Vorwärts-/Rückwärtszähler verwendet werden.

**Vergleichs-operationen**

Mit den Vergleichsoperationen werden die Inhalte der beiden AKKUs miteinander verglichen. Die AKKU-Inhalte werden dabei nicht verändert. Die einzelnen Operationen sind in der folgenden Tabelle aufgelistet; anschließend wird ihre Anwendung an einem Beispiel erläutert.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| !=F | | | **Vergleich auf gleich**<br>Die AKKU-Inhalte werden als Bitmuster interpretiert und auf Gleichheit abgefragt. |
| ><F | | | **Vergleich auf ungleich**<br>Die AKKU-Inhalte werden als Bitmuster interpretiert und auf Ungleichheit verglichen. |
| >F | | | **Vergleich auf größer**<br>Die AKKU-Inhalte werden als Festpunktzahlen interpretiert. Es wird untersucht, ob der Operand in AKKU 2 größer als der in AKKU 1 ist. |
| >=F | | | **Vergleich auf größer - gleich**<br>Die AKKU-Inhalte werden als Festpunktzahlen interpretiert. Die Abfrage untersucht, ob der Operand in AKKU 2 größer oder gleich dem Operanden in AKKU 1 ist. |
| <F | | | **Vergleich auf kleiner**<br>Die AKKU-Inhalte werden als Festpunktzahlen interpretiert. Es wird untersucht, ob der Operand in AKKU2 kleiner als der in AKKU 1 ist. |
| <=F | | | **Vergleich auf kleiner - gleich**<br>Die AKKU-Inhalte werden als Festpunktzahlen interpretiert. Die Abfrage untersucht, ob der Operand in AKKU 2 kleiner oder gleich dem Operanden in AKKU 1 ist. |

**Bearbeiten einer Vergleichsoperation**

Zum Vergleich zweier Operanden müssen diese nacheinander in die beiden AKKUs geladen werden.

Die Ausführung der Operation ist unabhängig vom VKE. Das Ergebnis ist binär und steht als VKE für die weitere Programmbearbeitung zur Verfügung. Ist der Vergleich erfüllt, so ist das VKE "1", anderenfalls ist es "0".

Bei der Ausführung der Vergleichsoperationen werden die Anzeigen gesetzt.

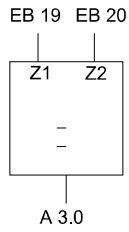
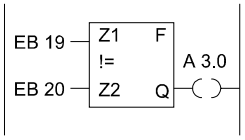
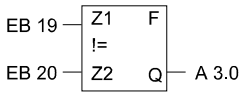
**Hinweis!**

Achten Sie auf gleiche Zahlenformate der Operanden.

**Vergleichs-operationen**

**Beispiel:**

*Vergleich auf gleich*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|  | L EB 19<br>L EB 20<br>! = F<br>= A 3.0 |  |  |

Der zuerst angegebene Operand wird mit dem nachfolgenden Operanden entsprechend der Vergleichsfunktion verglichen.

Der Vergleich ergibt ein binäres Verknüpfungsergebnis.

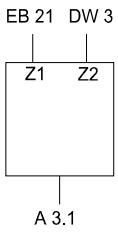
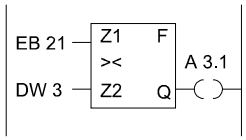
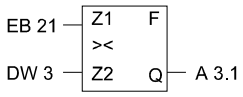
VKE = "1": Vergleich ist erfüllt, wenn AKKU-1 gleich AKKU-2.

VKE = "0": Vergleich ist nicht erfüllt, wenn AKKU-1 ungleich AKKU-2.

Die Anzeigen ANZ 1 und ANZ 0 werden, wie in der Operationsliste beschrieben, gesetzt.

Beim Vergleich wird die Zahlendarstellung der Operanden berücksichtigt, d.h. der Inhalt von AKKU-1 und AKKU-2 wird hier als Festpunktzahl interpretiert.

*Vergleich auf ungleich*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|  | L EB 21<br>L DW 3<br>> < F<br>= A 3.1 |  |  |

Der zuerst angegebene Operand wird mit dem nachfolgenden Operanden entsprechend der Vergleichsfunktion verglichen.

Der Vergleich ergibt ein binäres Verknüpfungsergebnis.
VKE = "1": Vergleich ist erfüllt, wenn AKKU-1 ungleich AKKU-2
VKE = "0": Vergleich ist nicht erfüllt, wenn AKKU-1 gleich AKKU-2

Die Anzeigen ANZ 1 und ANZ 0 werden, wie in der Operationsliste beschrieben, gesetzt.

Beim Vergleich wird die Zahlendarstellung der Operanden berücksichtigt, d.h. der Inhalt von AKKU-1 und AKKU-2 wird hier als Festpunktzahl interpretiert.

**Arithmetische Operationen**

Mit den arithmetischen Operationen werden die Inhalte der Akkumulatoren als Festpunktzahlen interpretiert und entsprechend der Rechenoperation miteinander verknüpft. Das Ergebnis wird im AKKU 1 hinterlegt. Die Operationen sind in der folgenden Tabelle aufgelistet und werden anschließend an einem Beispiel erläutert.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| +F | | | **Addieren**<br>Die Inhalte der beiden AKKUs werden addiert. |
| -F | | | **Subtrahieren**<br>Der Inhalt von AKKU 1 wird vom Inhalt des AKKU 2 subtrahiert. |

Für die Multiplikation und Division stehen in der CPU 24x Integrierte Funktionsbausteine zur Verfügung.

**Arithmetische Operationen**

**Bearbeitung einer Rechenoperation**

Vor Ausführung der arithmetischen Operation müssen die beiden Operanden in die AKKUs geladen werden.

**Hinweis!**
Achten Sie auf gleiche Zahlenformate der Operanden.

Die arithmetischen Operationen werden unabhängig vom VKE durchgeführt. Das Ergebnis steht im AKKU 1 für die Weiterverarbeitung zur Verfügung. Der Inhalt von AKKU 2 bleibt unverändert.

Die Operationen beeinflussen das VKE nicht; in Abhängigkeit vom Ergebnis werden die Anzeigen gesetzt.

| AWL | | Erläuterung |
|---|---|---|
| L | Z 3 | Der Wert von Zähler 3 wird in den AKKU 1 geladen. |
| L | Z 1 | Der Wert von Zähler 1 wird in den AKKU 1 geladen. Der vorherige Inhalt von AKKU 1 wird in den AKKU 2 "geschoben". |
| +F | | Die Inhalte der beiden AKKUs werden als 16-bit-Festpunktzahlen interpretiert und addiert. |
| T | AW 12 | Das Ergebnis - Inhalt von AKKU 1 - wird zum Ausgangswort 12 transferiert. |

Zahlenbeispiel

| | | | |
|---|---|---|---|
| 876 | 0 0 0 0 0 0 1 1 | 0 1 1 0 1 1 0 0 | AKKU 2 |
| + | | +F | |
| 668 | 0 0 0 0 0 0 1 0 | 1 0 0 1 1 1 0 0 | AKKU 1 |
| = | | | |
| 1544 | 0 0 0 0 0 1 1 0 | 0 0 0 0 1 0 0 0 | AKKU 1 |

**Bausteinoperationen**

Mit den Bausteinoperationen wird der Ablauf eines strukturierten Programmes festgelegt. Im Anschluss an die Übersicht werden die verschiedenen Operationen erklärt.

**Baustein-operationen**

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| SPA | **x** | **x** | **Sprung absolut**<br>Unabhängig vom VKE wird die Progammbearbeitung in einem anderen Baustein fortgesetzt.<br>Das VKE bleibt unbeeinflusst. |
| SPB | **x** | **x** | **Sprung bedingt**<br>Bei VKE "1" wird zu einem anderen Baustein gesprungen. Andernfalls wird die Programmbearbeitung im bisherigen Baustein fortgesetzt.<br>Das VKE wird dabei auf "1" gesetzt |
| Kennzeichen | | Parameter | |
| | OB | 0...255* | |
| | PB | 0...255 | |
| | FB | 0...255 | |
| | SB | 0...255 | |
| A | **x** | **x** | **Aufruf eines Datenbausteins**<br>Unabhängig vom VKE wird ein Datenbaustein aktiviert.<br>Die Programmbearbeitung wird nicht unterbrochen.<br>Das VKE bleibt unbeeinflusst. |
| E | **x** | **x** | **Erzeugen und Löschen eines Datenbausteins****<br>Unabhängig vom VKE wird ein Bereich im RAM-Speicher für die Ablage von Daten eingerichtet. |
| Kennzeichen | | Parameter | |
| | DB | 2...255*** | |
| BE | | | **Baustein beenden**<br>Unabhängig vom VKE wird der aktuelle Baustein beendet.<br>Die Programmbearbeitung wird im aufrufenden Baustein fortgesetzt. Das VKE wird "mitgenommen", aber nicht beeinflusst. |
| BEA | | | **Bausteinende absolut**<br>Unabhängig vom VKE wird der aktuelle Baustein beendet.<br>Die Programmbearbeitung wird im aufrufenden Baustein fortgesetzt. Das VKE wird "mitgenommen", aber nicht beeinflusst. |
| BEB | | | **Bausteinende bedingt**<br>Bei VKE "1" wird der aktuelle Baustein beendet.<br>Die Programmbearbeitung wird im aufrufenden Baustein fortgesetzt. Das VKE bleibt beim Bausteinwechsel unverändert "1". Bei VKE "0" wird die Operation nicht ausgeführt.<br>Das VKE wird auf "1" gesetzt und das Programm linear weiterbearbeitet. |

* Beim PG 615 muss bei der Voreinstellung "Systembefehle- Ja" eingestellt werden. Beachten Sie außerdem, dass vom Betriebssystem bestimmte OBs belegt sind.

** Die Länge des DB ist vor Ausführung des Befehls im AKKU 1 zu hinterlegen. Bei Länge 0 wird der DB ungültig.

*** Die Datenbausteine DB 0 und DB 1 sind reserviert.

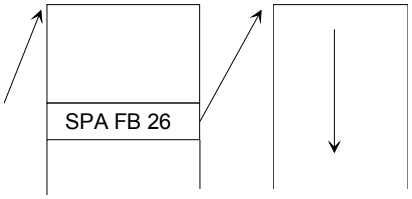
**Baustein-operationen**

**Absoluter Bausteinaufruf "SPA"**

Innerhalb eines Bausteines wird ein anderer Baustein aufgerufen, unabhängig von irgendwelchen Bedingungen.

**Beispiel:**

Im FB 26 wurde eine besondere Funktion programmiert. Sie wird an verschiedenen Stellen im Programm - z.B. im PB 63 - aufgerufen und bearbeitet.

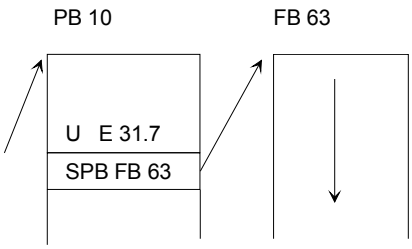
| Programmablauf | AWL | Erläuterung |
|---|---|---|
|  | .<br>.<br>.<br>.<br>.<br>SPA FB 26 | Die Anweisung "SPA FB 26" im Programmbaustein 63 bewirkt, dass der Funktionsbaustein 26 aufgerufen wird. |

**Bedingter Bausteinaufruf "SPB"**

Innerhalb eines Bausteins wird ein anderer Baustein aufgerufen, wenn die vorherige Bedingung erfüllt ist (VKE = 1).

**Beispiel:**

Im FB 63 wurde eine besondere Funktion programmiert, die unter bestimmten Voraussetzungen - z.B. im PB 10 - im Programm aufgerufen und bearbeitet wird.

| Programmablauf | AWL | Erläuterung |
|---|---|---|
|  | .<br>.<br>.<br>S M 1.0<br>U E 31.7<br>SPB FB63<br>.<br>. | Die Anweisung "SPB FB 63" im Programmbaustein 10 bewirkt, dass der Funktionsbaustein 63 aufgerufen wird, wenn der Eingang E 31.7 Signal "1" führt. |

**Baustein-operationen**

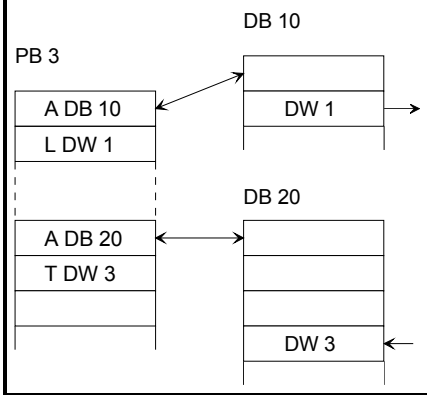
**Aufruf eines Datenbausteines "A DB"**

Datenbausteine werden immer absolut aufgerufen. Alle nachfolgenden Datenbearbeitungen beziehen sich auf den aufgerufenen Datenbaustein.

Mit dieser Operation können keine neuen Datenbausteine erzeugt werden. Die aufgerufenen Bausteine müssen vor der Programmbearbeitung programmiert oder erzeugt werden.

**Beispiel:**

Im Programmbaustein 3 wird eine Information benötigt, die im DB 10 als DW 1 programmiert wurde. Ein anderes Datum - z.B. ein Rechenergebnis - wird im DB 20 als DW 3 abgelegt.

| Programmablauf | AWL | Erläuterung |
|---|---|---|
|  | A DB 10<br>L DW 1<br>.<br>.<br>.<br>A DB 20<br>T DW 3 | Die Information aus dem Datenwort 1 im DB 10 wird in den Akkumulator geladen. Der Inhalt von AKKU 1 wird im Datenwort 3 von Datenbaustein 20 abgelegt |

**Erzeugen und Löschen eines Datenbausteins**

Die Anweisung "E DBx" ruft keinen DB auf, sondern erzeugt einen neuen Baustein. Sollen Daten aus diesem Datenbaustein verwendet werden, muss er mit der Anweisung A DB aufgerufen werden.

Vor "E DB" müssen Sie im AKKU 1 angeben, wie viele Datenwörter der Baustein umfassen soll.

Wird als Datenbausteinlänge Null angegeben, so wird der angegebene DB gelöscht, d.h. aus der Adressliste ausgetragen. Er gilt dann als nicht mehr vorhanden.

**Hinweis!**

Der Baustein bleibt so lange als "Leiche" erhalten, bis der AG-Speicher komprimiert wird.

Soll ein Datenbaustein eingerichtet werden, der bereits vorhanden ist, dann ist die Anweisung E DBx wirkungslos. Ist die Länge des einzurichtenden DBs größer als der zur Verfügung stehende Speicher, so geht die CPU mit "TRAF" in STOP bzw. verzweigt in den entsprechenden Fehlerreaktions-OB. Die Länge des eingerichteten Datenbausteins ist beliebig. Beachten Sie jedoch, dass die Programmiergeräte nur Bausteine mit einer begrenzten Länge bearbeiten können.

**Baustein-operationen**

**Erzeugen eines Datenbausteins**

| Beispiel | AWL | Erläuterung |
|---|---|---|
| Ohne Zuhilfenahme eines Programmiergerätes soll ein Datenbaustein mit 128 Datenworten erzeugt werden. | L KF +128<br>E DB 5 | Die konstante Festpunktzahl +128 wird in den AKKU1 geladen gleichzeitig wird der alte Inhalt von AKKU 1 in den AKKU 2 geschoben.<br>Der Datenbaustein 5 wird mit einer Länge von 128 Datenworten (0000) im RAM-Bereich des AGs erzeugt und in der Baustein-Adressliste eingetragen. Bei der nächsten Bearbeitung des Befehls E DB 5 bleibt dieser, falls der Inhalt von AKKU 1 nicht Null ist, wirkungslos. |

**Löschen eines Datenbausteins**

| Beispiel | AWL | Erläuterung |
|---|---|---|
| Ein nicht mehr benötigter Datenbaustein soll gelöscht werden | L KF +0<br>E DB 5 | Die konstante Festpunktzahl +0 wird in den AKKU 1 geladen, gleichzeitig wird der alte Inhalt von AKKU 1 in den AKKU 2 geschoben.<br>Der Datenbaustein 5 (er muss im RAM-Bereich des AGs stehen) wird für ungültig erklärt und aus der Bausteinadressliste ausgetragen. |

**Baustein-operationen**

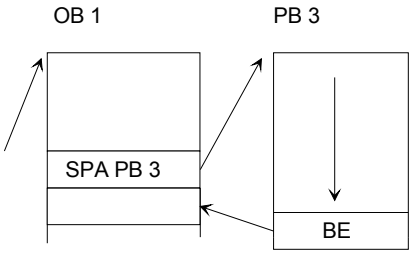
### Beenden eines Bausteins "BE"

Durch die Operation "BE" wird ein Baustein abgeschlossen; Datenbausteine brauchen nicht beendet zu werden. "BE" ist immer die letzte Anweisung eines Bausteins. Bei strukturierter Programmierung wird die Programmbearbeitung im aufrufenden Baustein fortgesetzt.

Binäre Verknüpfungen können im übergeordneten Baustein nicht fortgesetzt werden.

**Beispiel:**

Der Programmbaustein 3 wird durch die Anweisung "BE" beendet.

| Programmablauf | AWL | Erläuterung |
|---|---|---|
|  | .<br>.<br>.<br>.<br>.<br>BE | Die Anweisung "BE" beendet den PB 3 und bewirkt einen Rücksprung zum OB1. |

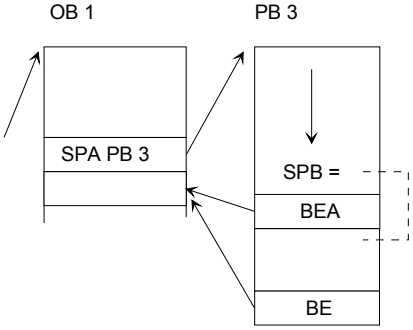
### Absoluter Rücksprung "BEA"

Die Operation "BEA" bewirkt einen Rücksprung innerhalb eines Bausteines. Sie kann jedoch in FBs durch Sprungoperationen umgangen werden.

Binäre Verknüpfungen können im übergeordneten Baustein nicht fortgesetzt werden.

Beispiel:

Die Bearbeitung des FB 21 wird ohne Rücksicht auf das VKE abgebrochen.

| Programmablauf | AWL | Erläuterung |
|---|---|---|
|  | .<br>.<br>.<br>.<br>SPB=<br>BEA<br>.<br>.<br>.<br>BE | Die Anweisung "BEA" bewirkt, dass der FB 21 verlassen wird. Es folgt ein Rücksprung auf den PB 8. |

**Baustein-operationen**

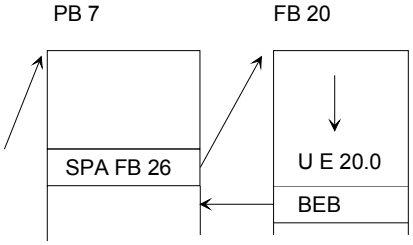
**Bedingter Rücksprung "BEB"**

Die Operation "BEB" bewirkt einen Rücksprung innerhalb eines Bausteines, wenn die vorherige Bedingung erfüllt ist (VKE=1).

Anderenfalls wird die lineare Bearbeitung des Programms mit VKE "1" fortgesetzt.

Beispiel:

Die Bearbeitung des FB 20 wird abgebrochen, wenn das VKE auf "1" ist.

| Programmablauf | AWL | Erläuterung |
|---|---|---|
|  | .<br>.<br>.<br>.<br>S A 1.0<br>U E 20.0<br>BEB<br>.<br>.<br>. | Die Anweisung "BEB" bewirkt einen Rücksprung vom FB 20 in den PB 7, wenn der Eingang E 20.0 Signal "1" führt. |

**Sonstige Operationen**

In der folgenden Tabelle werden weitere Grundoperationen aufgeführt und anschließend beschrieben.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| STP | | | **Stop** am Ende der Programmbearbeitung<br>Die aktuelle Programmbearbeitung wird zu Ende gebracht; das PAA wird ausgelesen. Dann geht das AG in STOP. |
| NOP 0 | | | **Nulloperation**<br>Im RAM-Speicher werden 16 Bits auf "0" gesetzt. |
| NOP 1 | | | **Nulloperation**<br>Im RAM-Speicher werden 16 Bits auf "1" gesetzt. |
| BLD | | **x** | **Bildaufbaubefehle** für das Programmiergerät |
| Kennzeichen | | Parameter<br>130, 131, 132, 133, 255 | |

**Hinweis!**

Diese Operationen können nur als AWL programmiert werden.

**STOP-Operation**

Durch die Operation "STP" wird das AG in den STOP-Zustand gebracht. Dies kann bei zeitkritischen Zuständen der Anlage oder bei Auftreten eines Gerätefehlers erwünscht sein.

Nach dem Bearbeiten der Anweisung wird das Steuerungsprogramm - ohne Berücksichtigung des VKE - bis zum Programmende abgearbeitet. Danach geht das AG mit der Fehlerkennung "STS" in STOP. Es kann dann über den Betriebsartenschalter (STOP→RUN) oder mit dem PG neu gestartet werden.

**Null-Operation**

Mit den Nulloperationen "NOP" werden Speicherplätze freigehalten oder überschrieben.

**Bildaufbauoperationen**

Innerhalb eines Bausteines werden Programmteile durch Bildaufbau-operationen "BLD" in Segmente unterteilt.

Die Null- und Bildaufbauoperationen sind nur für das PG bei der Darstellung des Programms von Bedeutung.

Die CPU führt bei der Bearbeitung dieser Anweisungen keine Operation aus.

# Ergänzende Operationen

Grundoperationen können in allen Bausteinen programmiert werden. Durch die "ergänzenden Operationen" wird der Operationsvorrat erweitert. Für diese Operationen gelten jedoch folgende Einschränkungen:

- Sie können nur in Funktionsbausteinen programmiert werden,
- Sie können nur als Anweisungsliste dargestellt werden.

In den folgenden Abschnitten werden die ergänzenden Operationen beschrieben.

**Ladeoperation**

Wie bei den Grundoperationen werden Informationen in den Akkumulator kopiert.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| L | **x** ↑ | **x** ↑ | **Laden**<br>Unabhängig vom VKE wird ein Wort aus den Systemdaten in den AKKU 1 geladen. |
| Kennzeichen<br>BS | | Parameter<br>0...255 | |

| Beispiel | AWL | Erläuterung |
|---|---|---|
| Im SD 103 wird bei Quittungsverzug die Fehleradresse abgelegt. Eine "wichtige" Ausgabebaugruppe ist unter der Anfangsadresse 4 gesteckt. Wird der Quittungsverzug durch diese Adresse ausgelöst, so soll die CPU in STOP verzweigen. Anderenfalls soll eine Meldung erfolgen, das Programm aber weiterbearbeitet werden. Dieses Beispiel können sie z.B. im OB24 programmieren. | L BS 103<br>L KH F004<br>.<br><> F<br>= A 12.0<br>BEB<br>.<br>STP | Der Inhalt des SD 103 und die Adresse der "wichtigen" Baugruppe werden in die AKKUs geladen.<br>Sind die beiden Werte verschieden, so wird A 12.0 gesetzt. Die Programmbearbeitung wird im OB1 (oder dem aufrufenden Baustein) fortgesetzt.<br>Wird beim Vergleich Gleichheit der Werte festgestellt, so verzweigt die CPU in STOP. |

**Freigabeoperation** Die Freigabeoperation "FR" wird dazu benutzt, um folgend Operationen auch ohne Flankenwechsel ausführen zu können:

- Starten einer Zeit
- Setzen eines Zählers
- Vor- und Rückwärtszählen.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| FR | **x** ↑ | **x** ↑ | **Freigabe einer Zeit/ eines Zähler**<br>Bei steigender Flanke des VKE werden Zeiten und Zähler freigegeben. Die Operation bewirkt den Neustart einer Zeit, das Setzen, Vorwärts- oder Rückwärtszählen eines Zählers, wenn an der "Startoperation" das VKE "1" anliegt. |
| Kennzeichen | | Parameter | |
| | T | 0...127 | |
| | Z | 0...127 | |

| Beispiel | AWL | Erläuterung |
|---|---|---|
| Eine Zeit T 2 wird durch E 2.5 als verlängerter Impuls gestartet (Impulsbreite 50s). Diese Zeit setzt den A 4.2 für die Dauer des Impulses. | U E 2.5<br>L KT 50.2<br>SV T 2<br>U T 2<br>= A 4.2<br>.<br>.<br>.<br>. | Starten einer Zeit T2 als verlängerter Impuls.<br>Der Ausgang 4.2 wird für 50s gesetzt. |
| Wird der A 3.4 immer wieder gesetzt, so soll die Zeit immer wieder von neuem gestartet werden | U A 3.4<br>FR T 2<br>.<br>BE | Wird der Ausgang 3.4 während der Zeit gesetzt (positiver Flankenwechsel des VKE), in der noch der Eingang 2.5 gesetzt ist, so wird die Zeit T2 neu gestartet.<br>Das heißt der A 4.2 bleibt um die erneut gestartete Zeit gesetzt, oder wird von neuem gesetzt.<br>Ist der E 2.5 beim Flankenwechsel von A 3.4 nicht gesetzt, so wird die Zeit nicht neu gestartet |

**Bit-Testoperation**

Mit den Bit-Testoperationen können digitale Operanden bitweise abgefragt und beeinflusst werden. Sie müssen immer am Beginn einer Verknüpfung stehen. Die folgende Tabelle gibt einen Überblick dieser Testoperationen.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| P | **x** | **x** | **Prüfe Bit auf Signalzustand "1"**<br>Unabhängig vom VKE wird ein einzelnes Bit abgefragt. Je nach dessen Signalzustand wird das VKE beeinflusst |
| PN | **x** | **x** | **Prüfe Bit auf Signalzustand "0"**<br>Unabhängig vom VKE wird ein einzelnes Bit abgefragt. Je nach dessen Signalzustand wird das VKE beeinflusst |
| SU | **x** | **x** | **Setzte Bit unbedingt**<br>Unabhängig vom VKE wird das angesprochene Bit auf "1" gesetzt. Das VKE wird nicht beeinflusst |
| RU | **x** | **x** | **Rücksetze Bit unbedingt**<br>Unabhängig vom VKE wird das angesprochene Bit auf "0" gesetzt. Das VKE wird nicht beeinflusst |
| Kennzeichen | | Parameter | |
| | T | 0...127.15 | |
| | Z | 0...127.15 | |
| | D | 0...255.15 | |
| | BS*1 | 0...255.15 | |

*1 nur bei P und PN

Die folgende Tabelle zeigt, wie das VKE bei den Bit-Testoperationen "P" und "PN" gebildet wird.

| Operation | P | | PN | |
|---|---|---|---|---|
| Signalzustand des Bits im angegebenen Operanden | 0 | 1 | 0 | 1 |
| Verknüpfungsergebnis | 0 | 1 | 1 | 0 |

**Bit Test-operationen**

| Beispiel | AWL | Erläuterung |
|---|---|---|
| Am Eingang E 2.0 ist eine Lichtschranke installiert, die Stückgut zählt. Nach jeweils 100 Stück soll entweder in den Funktionsbaustein FB 5 oder in den FB 6 verzweigt werden. Nach 800 Stück soll der Zähler 10 automatisch rückgesetzt werden und anschließend wieder hochzählen. | A DB 10<br>U E 2.0<br>ZV Z 10<br>U E 3.0<br>L KZ 0<br>S Z 10<br>O E 4.0<br>O M 5.2<br>R Z 10<br>LC Z 10<br>T DW 12<br><br>PN D 12.8<br><br>SPB FB 5<br><br>P D 12.8<br><br>SPB FB 6<br><br>P D 12.11<br><br>= M 5.2 | Aufruf des DB 10<br><br>Der Zählwert des Zähler Z10 wird durch den Eingang E 3.0 mit der Konstanten 0 geladen. Mit jedem Positiven Flankenwechsel am E2.0 wird der Zähler um 1 erhöht. Der Zähler wird entweder durch E 4.0 oder Merker M 5.2 zurückgesetzt. Der aktuelle Zählwert des Zählers wird BCD-codiert im DW 12 abgelegt.<br><br>Solange das Bit 8 des DW 12 Null ist, wird in den FB 5 gesprungen. Dies ist bei den ersten, dritten, fünften usw. hundert Stück der Fall.<br>Solange das Bit 8 des DW12 eins ist, wird in den FB 6 gesprungen. Dies ist bei den zweiten, vierten sechsten usw. hundert Stück der Fall.<br>Wenn das Bit 11 des DW12 eins wird (der Zählwert damit 800 ist) wird der Merker M 5.2 bedingt gesetzt. |
| Am Eingang E 10.0 ist eine Lichtschranke installiert, die Stückgut zählt. Nach jeweils 256 Stück soll der Zähler zurückgesetzt werden und von neuem hochzählen. | U E 10.0<br>ZV Z 20<br>U E 11.0<br>L KZ 0<br>S Z 20<br><br>P Z 20.8<br>SPB =VOLL<br>BEA<br>VOLL: RU Z 20.8<br>BE | Der Zählwert des Zählers 20 wird durch den E 11.0 mit der Konstanten 0 geladen. Mit jedem positiven Flankenwechsel am E 10.0 wird der Zählwert um 1 erhöht. Hat der Zählwert die Zahl 256 =100H erreicht (das Bit 8 ist "1"), so wird zu der Marke "VOLL" gesprungen, ansonsten wird der Baustein beendet.<br>Das Bit 8 des Zählers Z 20 wird unbedingt auf "0" gesetzt, damit steht im Zählwert wieder 000h. |

**Hinweis!**

Die Zeit und Zählwerte sind im Zähl-/Zeitwort hexadezimal in den 10 niederwertigsten Bits hinterlegt (Bit 0 bis Bit 9).

Die Zeitbasis (Zeitraster) ist in Bit 12 und Bit 13 des Zeitwortes hinterlegt.

**Wortweise Verknüpfungen**

Mit diesen Operationen werden die Inhalte der beiden AKKUs bitweise miteinander verknüpft. Die folgende Tabelle gibt eine Übersicht dieser Operationen, die dann an Beispielen erklärt werden

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| UW | | | Bitweise UND-Verknüpfung |
| OW | | | Bitweise ODER-Verknüpfung |
| XOW | | | Bitweise EXklusiv-ODER-Verknüpfung |

Bearbeitung einer Digitalverknüpfung

Die wortweisen Verknüpfungen werden unabhängig vom VKE ausgeführt. Umgekehrt beeinflussen sie das VKE nicht, aber die Anzeigen werden je nach "Rechenergebnis" gesetzt.

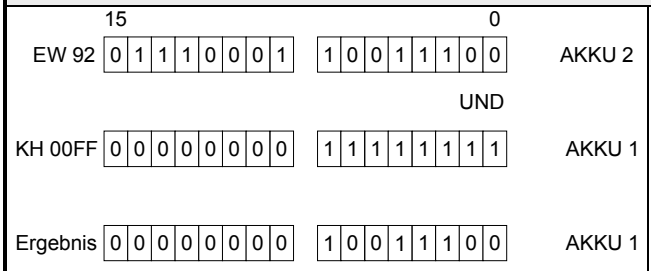
**Hinweis!**

Vor der Ausführung der Operationen müssen die beiden Operanden in die AKKUs geladen werden. Achten Sie dabei auf gleiche Zahlenformate!

**Wortweise Verknüpfungen**

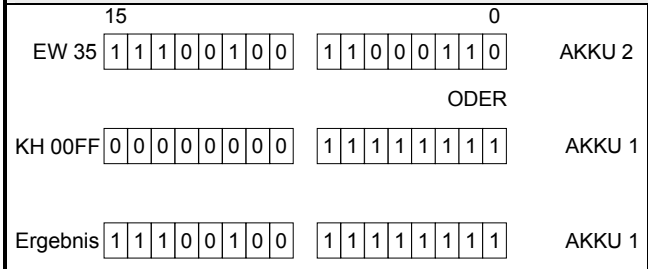
Das "Rechenergebnis" steht im AKKU 1 für die Weiterverarbeitung zur Verfügung. Der Inhalt von AKKU 2 bleibt unbeeinflusst.

| AWL | Erläuterung |
|---|---|
| L EW 92 | Das Eingangswort 92 wird in den AKKU 1 geladen |
| L KH 00FF | Ein konstanter Wert wird in den AKKU 1 geladen. Der vorherige Inhalt von AKKU 1 wird in den AKKU 2 "geschoben. |
| UW | Die Inhalte der beiden AKKUs werden Bit für Bit nach UND verknüpft |
| T AW 82 | Das Ergebnis - Inhalt von AKKU 1- wird zum Ausgangswort 82 transferiert |

| Zahlenbeispiel | |
|---|---|
|  | Im Eingangswort 92 sollen die 8 höherwertigen Bits auf "0" gesetzt werden.<br>Die beiden Wort werden bitweise verglichen. Steht in entsprechenden Bits eine "1", so wird das Ergebnisbit auf "1" gesetzt. |

**Wortweise Verknüpfungen** **OW**

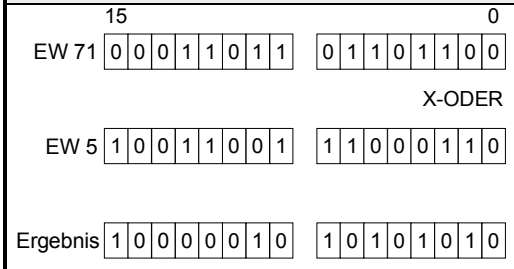
| AWL | Erläuterung |
|---|---|
| L EW 35 | Das Eingangswort 35 wird in den AKKU 1 geladen |
| L KH 00FF | Ein konstanter Wert wird in den AKKU 1 geladen. Der vorherige Inhalt von AKKU 1 wird in den AKKU 2 "geschoben. |
| OW | Die Inhalte der beiden AKKUs werden Bit für Bit nach ODER verknüpft |
| T EW 35 | Das Ergebnis - Inhalt von AKKU 1- wird zum Eingangswort 35 transferiert |

| Zahlenbeispiel | |
|---|---|
|  | Im Eingangswort 35 sollen die 8 niederwertigen Bits auf "1" gesetzt werden.<br>Im Ergebniswort wird eine "1" gesetzt, wenn in den entsprechenden Bits der beiden Wort eine "1" steht. |

**Wortweise Verknüpfungen** **XOW**

| AWL | Erläuterung |
|---|---|
| L EW 71 | Das Eingangswort 71 wird in den AKKU 1 geladen |
| L EW 5 | Das Eingangswort 5 wird in den AKKU 1 geladen. Der vorherige Inhalt von AKKU 1 wird in den AKKU 2 "geschoben. |
| XOW | Die Inhalte der beiden AKKUs werden Bit für Bit nach Exklusiv-ODER verknüpft |
| T AW 86 | Das Ergebnis - Inhalt von AKKU 1 - wird zum Ausgangswort 86 transferiert |

Zahlenbeispiel




Die Eingangsworte 71 und 5 sollen auf Gleichheit überprüft werden.

Das Ergebnisbit wird nur dann auf "1 gesetzt, wenn in AKKU1 und AKKU 2 unterschiedliche Bits stehen.

**Schiebe-operationen**

Mit diesen Operationen wird das Bitmuster in AKKU 1 verschoben; der Inhalt von AKKU2 bleibt unverändert. Durch das Verschieben erreicht man eine Multiplikation oder Division des Inhalts von AKKU1 mit 2er-Potenzen.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| SLW | | **x** | **Schieben nach links**<br>Das Bitmuster im AKKU 1 wird nach links verschoben |
| SRW | | **x** | **Schieben nach rechts**<br>Das Bitmuster im AKKU1 wird nach rechts verschoben |
| | | ↑<br>Parameter<br>0...15 | |

**Bearbeitung einer Schiebeoperation**

Die Ausführung der Schiebeoperationen ist unabhängig von Bedingungen. Das VKE wird nicht beeinflusst. Durch Schiebeoperationen werden aber die Anzeigen gesetzt.

Der Zustand des zuletzt hinausgeschobenen Bits kann deshalb mit Sprungfunktionen abgefragt werden.

Der Parameter der Anweisung gibt die Anzahl der Bitstellen an, um die der Inhalt von AKKU 1 nach links (SLW) oder nach rechts (SRW) verschoben wird. Die beim Schieben frei werdenden Bitstellen werden mit Nullen aufgefüllt.

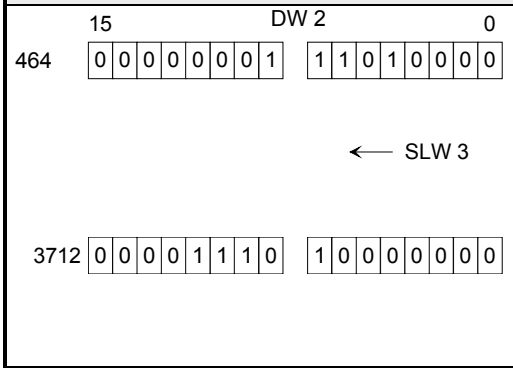
Der Inhalt der "hinausgeschobenen" Bits geht verloren. Der Zustand des Bits $2^0$ (SRW) oder des Bits $2^{15}$ (SLW) beeinflusst nach Ausführung des Befehls das ANZ 1-Bit. Dieses Bit kann ausgewertet werden.

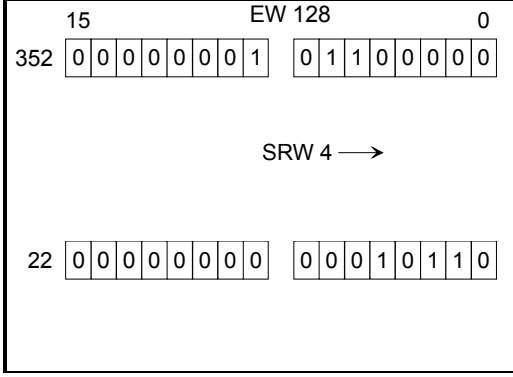
Eine Schiebeoperation mit dem Parameter "0" wird wie eine Null-Operation (NOP) behandelt. Der Zentralprozessor bearbeitet ohne weitere Reaktion die nächste Anweisung.

Vor der Ausführung der Operationen muss der Operand, der bearbeitet werden soll, in den AKKU 1 geladen werden.

Der veränderte Operand steht dort für die weiter Verarbeitung zur Verfügung.

**Schiebe-
operationen**

| AWL | Erläuterung |
|---|---|
| L DW 2 | Der Inhalt des Datenwortes 2 wird in den AKKU 1 geladen |
| SLW 3 | Das Bitmuster im AKKU 1 wird um drei Stellen nach links geschoben |
| T DW 3 | Das Ergebnis - Inhalt von AKKU 1- wird zum Datenwort 3 transferiert |
| Zahlenbeispiel | |
|  | Im Datenwort 2 ist der Wert 464 gespeichert. Dieser Wert soll mit $2^3$=8 multipliziert werden. dazu wird das Bitmuster von DW 2 im AKKU 1 um drei Stellen nach links geschoben |

| AWL | Erläuterung |
|---|---|
| L EW 128 | Der Inhalt des EW 128 wird in den AKKU 1 geladen |
| SRW 4 | Das Bitmuster im AKKU 1 wird um vier Stellen nach rechts geschoben |
| T AW 160 | Das Ergebnis - Inhalt von AKKU 1- wird zum AW 160 transferiert |
| Zahlenbeispiel | |
|  | Das EW 128 liefert den Wert 352 gespeichert. Wird im AKKU 1 das entsprechende Bitmuster um vier Stellen nach rechts geschoben, so wird der Wert 352 durch $2^4$ = 16 dividiert. |

**Umwandlungs-operationen**

Mit diesen Operationen können Sie die Werte im AKKU 1 umwandeln.

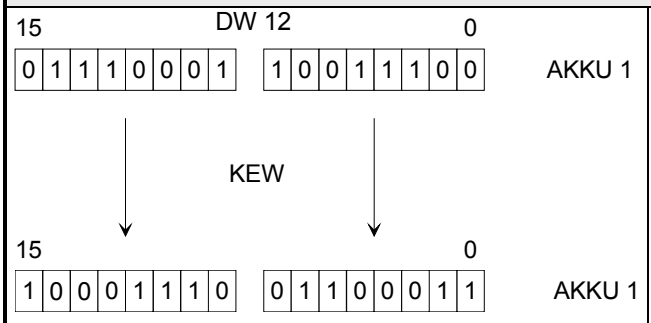
| Operation | Operand | | Bedeutung |
|---|---|---|---|
| KEW | | | **1er-Komplement**<br>Der Inhalt von AKKU 1 wird bitweise invertiert. |
| KZW | | | **2er-Komplement**<br>Der Inhalt von AKKU 1 wird bitweise invertiert. Anschließend wird der Wort 0001h addiert. |

**Bearbeitung der Umwandlungsoperationen**

Die Ausführung dieser Operationen hängt weder von VKE ab, noch hat es einen Einfluss darauf. Durch die Operation "KZW" werden die Anzeigen gesetzt.

| AWL | Erläuterung |
|---|---|
| L DW 12 | Der Inhalt des Datenwortes 12 wird in den AKKU 1 geladen |
| KEW | Alle Bits im AKKU 1 werden invertiert. |
| T AW 20 | Das Ergebnis - Inhalt von AKKU 1- wird zum AW 20 transferiert |

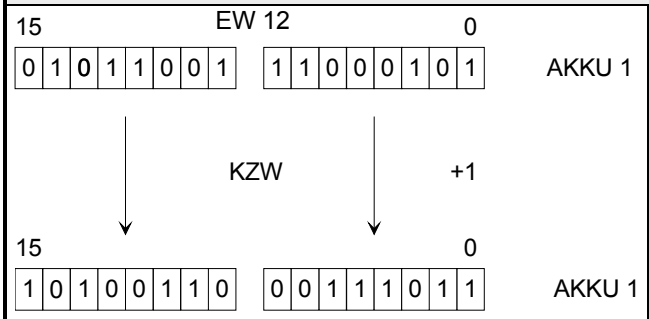
Zahlenbeispiel




In einer Anlage wurden Schließer durch Öffner ersetzt. Soll die Information im DW 12 die bisherige Auswirkung behalten, so muss das DW 12 invertiert werden.

**Umwandlungs-
operationen**

| AWL | Erläuterung |
|---|---|
| L EW 12 | Der Inhalt des EW12 wird in den AKKU 1 geladen. |
| KZW | Alle Bits im AKKU 1 werden invertiert, an der niedrigsten Stelle wird eine "1" addiert. |
| T DW 100 | Das veränderte Wort wird ins DW 100 transferiert. |

| Zahlenbeispiel | |
|---|---|
|  | Vom Wert des EW 12 soll der negative Wert gebildet werden. |

**Dekrementieren / Inkrementieren**

Mit diesen Operationen werden in dem AKKU 1 geladene Daten verändert.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| D | | **x** | **Dekrementieren**<br>Der Akkumulatorinhalt wird erniedrigt. |
| I | | **x** | **Inkrementieren**<br>Der Akkumulatorinhalt wird erhöht.<br>Der Inhalt von AKKU 1 wird um die im Parameter angegebene Zahl inkrementiert.<br>Die Operationsausführung ist unabhängig von Bedingungen.<br>Sie beschränkt sich auf das rechte Byte (ohne Übertrag). |
| | | Parameter<br>0...15 | |

**Bearbeitung**

Die Ausführung dieser beiden Operationen ist unabhängig vom VKE. Sie beeinflusst ihrerseits weder das VKE noch die Anzeigen.

Mit dem Parameter geben Sie an, um welchen Wert der Inhalt des AKKU 1 verändert werden soll. Die Operationen beziehen sich auf dezimale Werte; das Ergebnis wird jedoch dual im AKKU 1 hinterlegt.

Die Veränderungen beziehen sich außerdem nur auf das Low-Byte im Akkumulator.

| Beispiel | AWL | Erläuterung |
|---|---|---|
| Die Hexkonstante 1010h soll um das Inkrement 16 erhöht werden und im Datenwort 8 abgelegt werden. | A DB 6 | Aufruf des DB 6. |
| | L KH 1010 | Lade Hexkonstante 1010h in den AKKU 1. |
| | I 16 | Inkrementiere das Low-Byte von AKKU 1 um 16. Das Ergebnis 1020h befindet sich im AKKU 1. |
| | T DW 8 | Transferiere den Inhalt von AKKU 1 (1020h) in das DW 8. |
| Außerdem soll das Ergebnis vom Inkrementieren um das Dekrement 33 erniedrigt werden und im DW 9 abgelegt werden. | | Da im AKKU 1 noch das Ergebnis vom Inkrementieren steht, kann man direkt das Dekrement 33 davon bilden |
| | D 33 | das Ergebnis wäre FFFh. Da aber das High-Byte des AKKU 1 nicht dekrementiert wird, steht im AKKU 1 10FFh als Ergebnis |
| | T DW 9 | Der Inhalt von AKKU 1 wird ins DW 9 transferiert (10FFh). |

**Alarme sperren/freigeben**

Diese Operationen beeinflussen die alarm- und zeitgesteuerte Programmbearbeitung. Sie verhindern, dass die Bearbeitung einer Reihe von Anweisungen oder Bausteinen durch Prozess- oder Zeitalarme unterbrochen wird.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| AS | | | **Alarm sperren** |
| AF | | | **Alarm freigeben** |

**Bearbeitung**

Die Ausführung dieser Operationen ist vom VKE nicht abhängig. Sie haben ihrerseits keinen Einfluss auf das VKE und die Anzeigen. Nach Bearbeitung der Anweisung "AS" werden keine Alarme mehr ausgeführt. Die Anweisung "AF" hebt diese Wirkung wieder auf.

| Beispiel | AWL | Erläuterung |
|---|---|---|
| Alarmbearbeitung in einem bestimmten Programmteil sperren und dann wieder freigeben. | .<br>.<br>.<br>.<br>= A 7.5<br>AS<br>U E 2.3<br>.<br>.<br>SPA FB 3<br>.<br>.<br>.<br>AF<br>.<br>. | <br><br><br><br><br>**Alarm sperren**<br><br><br><br>Tritt ein Alarm auf, so wird der Programmabschnitt zwischen den Befehlen "AS" und "AF" unverzögert bearbeitet.<br><br>**Alarm freigeben**.<br>Inzwischen aufgetretene Alarme werden nach dem Befehl "AF" bearbeitet. |

**Bearbeitungs-operation**

Mit der Operation "B" können Anweisungen "indiziert" bearbeitet werden. Sie haben dadurch die Möglichkeit, den Parameter eines Operanden während der Bearbeitung des Steuerungsprogramms zu ändern.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| B | **x** ↑ | **x** ↑ | **Bearbeiten eines Merker- oder Datenwortes** |
| Kennzeichen | | Parameter | |
| | MW | 0...254 | |
| | DW | 0...255 | |

**Bearbeitung**

Die Anweisung "Bearbeite Merker- oder Datenwort x" ist eine 2-Wort-Anweisung, die unabhängig vom VKE ausgeführt wird.

Sie besteht genauer gesagt aus zwei zusammengehörenden Anweisungen:

- In der ersten Anweisung steht die Bearbeitungsoperation und die Angabe eines Merker- oder Datenwortes.
- In der zweiten Anweisung legen Sie die Operation und das Operandenkennzeichen fest, die vom Steuerungsprogramm bearbeitet werden sollen. Als Parameter müssen Sie hier 0 oder 0.0 eingeben.

Das Steuerungsprogramm arbeitet dann mit dem Parameter, der in dem Merker- oder Datenwort abgelegt ist, das von der ersten Anweisung aufgerufen wurde. Sollen binäre Operationen, Eingänge, Ausgänge oder Merker indiziert werden, so geben Sie im High-Byte dieses Wortes die Bitadresse, und im Low-Byte die Byteadresse an.

In allen anderen Fällen muss der High-Byte "0" sein.

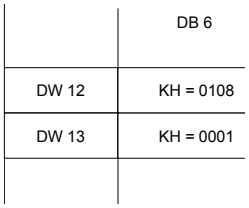
**Bearbeitungs-anweisungen**

Folgende Operationen können mit der Bearbeitungsanweisung kombiniert werden:

| Operation | Erläuterung |
|---|---|
| U *1,UN, O,ON | Binäre Verknüpfungen |
| S, R, = | Speicheroperationen |
| FR T, RT, SA T, SE T,SI T, SS T, SV T | Zeitoperationen |
| FR Z, RZ, SZ, ZR Z,ZV Z | Zähloperationen |
| L, LC, T | Lade- und Transferoperationen |
| SPA=,SPB=,SPZ=,SPN=,SPP=,SPM=,SPO= | Sprungoperationen |
| SLW, SRW | Schiebeoperationen |
| D, I | De- und Inkrementieren |
| A DB, SPA, SPB, TNB | Bausteinaufrufe |

*1 Die Operation "UE" wird in Kombination mit "B DW" oder "B MW" zur Operation "UA", wenn die Byteadresse im Daten oder Merkerwort größer als 127 ist.

Das folgende Bild zeigt, wie durch den Inhalt eines Datenwortes der Parameter der nächsten Anweisung bestimmt wird.




| FBx | ausgeführtes Programm |
|---|---|
| :A DB 6 | A DB 6 |
| . | . |
| :B DW 12 | |
| :U E 0.0 | U E 8.1 |
| :B DW 13 | |
| :FR T 0 | FR T 1 |

**Bearbeitungs-operationen**

Das folgende Beispiel zeigt, wie bei jeder Programmbearbeitung neue Parameter erzeugt werden.

| Beispiel | AWL | Erläuterung |
|---|---|---|
| Es sollen die Inhalte der Datenwörter DW 20 bis DW100 auf Signalzustand "0" gesetzt werden. Das "Index-register" für den Parameter der Datenwörter in DW 1. | :A DB 202 | Aufruf Datenbaustein 202. |
| | L KB 20 | Lade konstante Zahl 20 in AKKU1. |
| | T DW 1 | Transferiere Inhalt von AKKU1 ins Datenwort 1. |
| | M 1.. .L KH 0 | Lade Hexkonstante 0 AKKU1. |
| | B DW 1 | Bearbeite Datenwort 1. |
| | T DW 0 | Transferiere den Inhalt von AKKU 1 in das Datenwort, dessen Adresse im Datenwort 1 hinterlegt ist. |
| | L DW 1 | Lade Datenwort 1 in den AKKU1. |
| | L KB 1 | Lade konstante Zahl 1 in AKKU1. |
| | | Datenwort 1 wird in AKKU 2 geschoben. |
| | +F | AKKU 2 und AKKU 1 werden addiert und das Ergebnis in AKKU 1 hinterlegt (Erhöhung der Datenwortadresse). |
| | T DW 1 | Transferiere Inhalt von AKKU 1 ins Datenwort 1 (neue Datenwortadresse). |
| | L KB 100 | Die konstante Zahl 100 wird in AKKU1 geladen und die neue Datenwortadresse in AKKU 2 geschoben. |
| | <=F | Vergleich der AKKUs auf kleiner AKKU 2 <=AKKU1. |
| | SPB =M1 | Springe bedingt zur Marke M1, solange AKKU 2 <= AKKU 1 ist. |

**Sprung-operationen**

Die verschiedenen Operationen sind in der folgenden Tabelle aufgeführt Ein Beispiel zeigt, wie Sie Sprungoperationen einsetzen können.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| SPA= | **x** | | **Sprung absolut**<br>Der unbedingte Sprung wird unabhängig von Bedingungen ausgeführt. |
| SPB= | **x** | | **Sprung bedingt**<br>Der bedingte Sprung wird ausgeführt, wenn das VKE "1" ist. Bei VKE "0" wird die Anweisung nicht ausgeführt und das VKE auf "1" gesetzt. |
| SPZ= | **x** | | **Springe bei Ergebnis "Null(Zero)"**<br>Der Sprung wird nur ausgeführt, wenn ANZ 1 = 0 und ANZ 0 =0. Das VKE wird nicht verändert. |
| SPN= | **x** | | **Springe bei "Nicht Null"**<br>Der Sprung wird nur ausgeführt, wenn ANZ 1 ungleich ANZ 0 ist. Das VKE wird nicht verändert. |
| SPP= | **x** | | **Springe bei positivem Ergebnis**<br>Der Sprung wird nur ausgeführt, wenn ANZ 1 =1 und ANZ 0 =0 ist Das VKE wird nicht verändert. |
| SPM= | **x** | | **Springe bei negativem Ergebnis**<br>Der Sprung wird nur ausgeführt, wenn ANZ 1 =0 und ANZ 0 = 1. Das VKE wird nicht verändert. |
| SPO= | **x** | | **Sprung bei Überlauf (Overflow)**<br>Der Sprung wird ausgeführt, wenn ein Überlauf vorliegt. Andernfalls wird der Sprung nicht ausgeführt. Das VKE wird nicht verändert. |
| Kennzeichen Sprungmarke (max. 4 Zeichen) | | | |

**Bearbeitung der Sprungoperationen**

Neben der Sprungoperation muss immer ein symbolisches Sprungziel (Sprungmarke) eingegeben werden, das aus maximal vier Zeichen bestehen darf. Dabei muss das erste Zeichen ein Buchstabe sein.

Bei der Programmierung muss berücksichtigt werden:

- Die absolute Sprungdistanz darf nicht mehr als +127 oder -128 Wörter im Programmspeicher betragen. Beachten Sie, dass einige Anweisungen (z.B. "Laden einer Konstanten") zwei Worte belegen. Bei größeren Sprünge muss ein "Zwischenziel" eingefügt werden.
- Sprünge dürfen nur innerhalb eines Bausteins durchgeführt werden.
- Segmentgrenzen ("BLD 255") dürfen nicht übersprungen werden.

**Sprung-operationen**

| Beispiel | AWL | Erläuterung |
|---|---|---|
| Ist kein Bit des Eingangswortes 1 gesetzt, so wird zur Marke "AN 1" gesprungen. Stimmen Eingangswort 1 und Ausgangswort 3 nicht überein, so wird zur Marke "AN 0"zurückgesprungen. Andernfalls wird EW 1 mit dem Datenwort 12 verglichen. Ist EW 1 größer oder kleiner als DW12, so wird zur Marke "Ziel" gesprungen. | AN0 :L EW 1<br>:L KH0000<br>:+F<br>:SPZ =AN1<br>:U E 1.0<br>.<br>.<br>.<br>AN1 :L EW 1<br>:L AW 3<br>:XOW<br>:SPN =AN0<br>:L EW 1<br>:L DW 12<br>:><F<br><br><br><br>:SPB =ZIEL<br>.<br>.<br>.<br>ZIEL :U E 12.2 | Das Eingangswort 1 wird in den AKKU 1 geladen. Ist der Inhalt von AKKU 1 gleich Null *1, so wird zur Marke "AN1"gesprungen, andernfalls wird die nächste Anweisung (UE 1.0) bearbeitet.<br><br><br><br>Vergleich von Eingangswort 1 und Ausgangswort 3. Bei Ungleichheit sind einzelne Bits im AKKU 1 gesetzt.<br>Ist der Inhalt von AKKU 1 nicht Null, so wird zur Marke "AN 0" zurückgesprungen. Im anderen Fall werden die nächsten Anweisungen bearbeitet. Das EW 1 wird mit dem DW 12 verglichen. Bei Ungleichheit wird VKE "1" gesetzt Bei VKE = "1", wird zur Marke "ZIEL" gesprungen.<br>Ist "VKE" = "0" so wird die nächste Anweisung bearbeitet. |

*1 Die Anweisung "L.." hat keinen Einfluss auf die Anzeigen. Damit der AKKU-Inhalt mit der Operation "SPZ=" ausgewertet werden kann, wird eine Addition (+F) mit der Konstanten 0000h durchgeführt.

**Substitutions-operationen**

Soll ein Programm ohne größere Veränderung mit verschiedenen Operanden bearbeitet werden, so ist es zweckmäßig, die einzelnen Operanden zu parametrieren.

Müssen Operanden geändert werden, so brauchen nur die Parameter im Funktionsbaustein-Aufruf neu belegt zu werden.

Im Programm werden diese Parameter als "Formaloperanden" bearbeitet. Dazu sind besondere Operationen notwendig, die sich in ihrer Auswirkung jedoch nicht von der Operationen ohne Substitution unterscheiden. Auf den folgenden Seiten finden Sie eine kurze Beschreibung dieser Operationen mit passenden Beispielen.

### Binäre Verknüpfungen

| Operation | Operand | | Bedeutung | | |
|---|---|---|---|---|---|
| U = | | **x** | **UND-Verknüpfung**<br>Abfrage eines Formaloperanden auf Signalzustand "1" | | |
| UN = | | **x** | **UND-Verknüpfung auf Null**<br>Abfrage eines Formaloperanden auf Signalzustand "0" | | |
| O = | | **x** | **ODER-Verknüpfung**<br>Abfrage eines Formaloperanden auf Signalzustand "1" | | |
| ON = | | **x** ↑ | **ODER-Verknüpfung auf Null**<br>Abfrage eines Formaloperanden auf Signalzustand "0" | | |
| Formaloperanden | | | zulässige Aktualoperanden | Parameter Art | Parameter Typ |
| | | | binär adressierte Ein-, Ausgänge und Merker<br>Zeiten und Zähler | E, A<br><br>T, Z | BI |

**Substitutions-operationen**

**Speicheroperationen**

Die einzelnen Operationen werden in der folgenden Tabelle aufgezählt und anschließend durch ein Beispiel erläutert.

| Operation | Operand | | Bedeutung | | |
|---|---|---|---|---|---|
| S = | | **x** | Setzen (binär) eines Formaloperanden | | |
| RB = | | **x** | Rücksetzen (binär) eines Formaloperanden. | | |
| = = | | **x** ↑ | Zuweisen:<br>Das VKE wird einem Formaloperanden zugewiesen. | | |
| Formaloperanden | | | zulässige Aktualoperanden | Parameter Art | Parameter Typ |
| | | | binär adressierte Ein-, Ausgänge und Merker | E, A | BI |

**Beispiel:**

Im OB 1 wird der FB 30 parametriert:

| Aufruf in OB 1 | | Programm im FB 30 | | ausgeführtes Programm | |
|---|---|---|---|---|---|
| | :SPA FB 30 | :U | =EIN 1 | :U | E 2.0 |
| NAME | :VERKNUE | :UN | =EIN2 | :UN | E 2.1 |
| EIN1 | :E 2.0 | :O | =EIN3 | :O | E 2.2 |
| EIN2 | :E 2.1 | :S | =MOT5 | :S | A 7.3 |
| EIN3 | :E 2.2 | := | =AUS1 | := | A 7.1 |
| VEN1 | :E 2.3 | :U | =VEN1 | :U | E 2.3 |
| AUS1 | :A 7.1 | :U | =EIN2 | :U | E 2.1 |
| AUS2 | :A 7.2 | :ON | =EIN3 | :ON | E 2.2 |
| MOT5 | :A 7.3 | :RB | =MOT5 | :R | A 7.3 |
| | :BE | := | =AUS2 | := | A 7.2 |
| | | :BE | | :BE | |

**Substitutions-operationen**

**Lade- und Transferoperationen**

Die verschiedenen Operationen werden in der folgenden Tabelle aufgezählt und in einem Beispiel beschrieben.

| Operation | Operand | | Bedeutung | | |
|---|---|---|---|---|---|
| L = | | **x** | Laden eines Formaloperanden | | |
| LC = | | **x** | Laden codiert eines Formaloperanden | | |
| LW = | | **x** | Laden des Bitmusters eines Formaloperanden | | |
| T = | | **x** ↑ | Transferieren zu einem Formaloperanden | | |
| Formaloperanden | | | zulässige Aktualoperanden | Parameter Art | Parameter Typ |
| für L = | | | byte- und wort-adressierte Ein-, Ausgänge und Merker *1<br>Zeiten und Zähler | E,A<br><br><br>T,Z | BY,W |
| für LC = | | | Zeiten und Zähler | T,Z | |
| für LW = | | | Bitmuster | D | KF, KH, KM, KY, KC, KT, KZ |
| für T = | | | byte- und wort-adressierte Ein-, Ausgänge und Merker *1 | E,A | BY,W |

*1 Datenwort: DW, DR, DL

**Substitutions-operationen**

**Beispiel:** Im PB 1 wird der FB34 parametriert:

| Aufruf in PB 1 | | Programm im FB 34 | | ausgeführtes Programm | |
|---|---|---|---|---|---|
| | :SPA FB 34 | :U | =E0 | :U | E 2.0 |
| NAME | :LADE/TRAN | :L | =L1 | :L | MW 10 |
| E0 | :E 2.0 | :S | Z 6 | :S | Z 6 |
| E1 | :E2.1 | :U | =E1 | :U | E 2.1 |
| L1 | :MW 10 | :LW | =LW1 | :L | KZ 140 |
| LW1 | :KZ 140 | :S | Z 7 | :S | Z 7 |
| LC1 | :Z7 | :U | E 2.2 | :U | E 2.2 |
| T1 | :AW 4 | :ZV | Z 6 | :ZV | Z 6 |
| LW2 | :KZ 160 | :ZV | Z 7 | :ZV | Z 7 |
| | :BE | :LC | =LC1 | :LC | Z 7 |
| | | :T | =T1 | :T | AW 4 |
| | | :U | E 2.7 | :U | E 2.7 |
| | | :R | Z 6 | :R | Z 6 |
| | | :R | Z 7 | :R | Z 7 |
| | | :LW | =LW2 | :L | KZ 160 |
| | | :LC | =LC1 | :LC | Z 7 |
| | | :!=F | | :!=F | |
| | | :R | Z 7 | :R | Z 7 |
| | | :BE | | :BE | |

**Substitutions-operationen**

**Zeit- und Zählloperationen**

In der folgenden Tabelle werden die einzelnen Operationen aufgelistet. Anhand einiger Beispiele wird ihre Bedeutung erklärt.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| FR = | | x | **Freigabe** eines Formaloperanden für Neustart (Beschreibung siehe "FT" oder "FZ", je nach Formaloperand). |
| RD = | | x | **Rücksetzen (digital**) eines Formaloperanden. |
| SI = | | x | Starten einer als Formaloperand vorgegebenen Zeit mit dem im AKKU hinterlegten Wert als Impuls. |
| SE = | | x | **Starten** einer als Formaloperand vorgegebenen Zeit mit dem im AKKU hinterlegten Wert als Einschaltverzögerung. |
| SVZ = | | x | **Starten** einer als Formaloperand vorgegebenen Zeit mit dem im AKKU hinterlegten Wert als verlängerter Impuls bzw.. **Setzen** eines als Formaloperand vorgegebenen Zählers mit dem im AKKU angegebenen Zählwertes. |
| SSV = | | x | **Starten** einer als Formaloperand vorgegebenen Zeit mit dem im AKKU hinterlegten Wert als speichernde Einschalt-verzögerung bzw. **Vorwärtszählen** eines als Formal operand vorgegebenen Zählers. |
| SAR = | | x | **Starten** einer als Formaloperand vorgegebenen Zeit mit dem im AKKU hinterlegten Wert als Ausschaltverzögerung bzw. **Rückwärtszählen** eines als Formaloperand vorgegebenen Zählers. |

| Formaloperanden | zulässige Aktualoperanden | Parameter Art | Parameter Typ |
|---|---|---|---|
| | Zeiten und Zähler *1 | T,Z *1 | |

*1 nicht bei "SI" und "SE"

**Vorgabe der Zeit- und Zählwerte:**

Der Zeit- oder Zählwert kann wie bei den Grundoperationen als Formal-operand vorgegeben werden. In diesem Fall muss unterschieden werden, ob der Wert in einem Operandenwort liegt oder als Konstante angegeben wird.

- Operandenworte können die Parameterart E oder A und den Typ W haben. Sie werden mit der Operation "L =" in den AKKU geladen.
- Bei einer Konstanten ist die Parameterart "D", der Typ kann "KT" oder "KZ" sein. Diese Formaloperanden werden mit "LW =" in den AKKU geladen.

**Substitutions-operationen**

Die folgenden Beispiele zeigen, wie Sie mit den Zeit- und Zähloperationen arbeiten können.

**Beispiel 1:**

| Aufruf des FB | | Programm im FB 32 | | ausgeführtes Programm | |
|---|---|---|---|---|---|
| | :SPA FB 32 | :UN | =E5 | :UN | E 2.5 |
| NAME | :ZEIT | :U | =E6 | :U | E 2.6 |
| E5 | :E 2.5 | :L | KT 5.2 | :L | KT 5.2 |
| E6 | :E 2.6 | :SAR | =ZEI5 | :SA | T 5 |
| ZEI5 | :T 5 | :U | =E5 | :U | E 2.5 |
| ZEI6 | :T 6 | :UN | =E6 | :UN | E 2.6 |
| AUSG | :A 7.6 | :L | KT 5.2 | :L | KT 5.2 |
| | :BE | :SSV | =ZEI6 | :SS | T 6 |
| | | :U | =ZEI5 | :U | T 5 |
| | | :O | =ZEI6 | :O | T 6 |
| | | := | =AUS6 | := | A 7.6 |
| | | :U | E 2.7 | :U | E 2.7 |
| | | :RD | =ZEI5 | :R | T 5 |
| | | :RD | =ZEI6 | :R | T 6 |
| | | :BE | | :BE | |

**Beispiel 2:**

| Aufruf des FB | | Programm im FB 33 | | ausgeführtes Programm | |
|---|---|---|---|---|---|
| | :SPA FB 33 | :U | =E2 | :U | E 2.2 |
| NAME | :ZAEHL | :L | KZ 17 | :L | KZ 17 |
| E2 | :E2.2 | :SVZ | =ZAE5 | :S | Z 5 |
| E3 | :E 2.3 | :U | =E3 | :U | E 2.3 |
| E4 | :E2.4 | :SSV | =ZAE5 | :ZV | Z 5 |
| ZAE5 | :Z5 | :U | =E4 | :U | E 2.4 |
| AUS3 | :A7.3 | :SAR | =ZAE5 | :ZR | Z 5 |
| | :BE | :U | =ZAE5 | :U | Z 5 |
| | | := | =AUS3 | := | A 7.3 |
| | | :U | E 2.7 | :U | E 2.7 |
| | | :RD | =ZAE5 | :R | Z 5 |
| | | :BE | | :BE | |

**Substitutions-operationen**

**Bearbeitungsoperation**

Durch die Tabelle und ein Beispiel wird diese Operation erklärt:

| Operation | Operand | | Bedeutung | | |
|---|---|---|---|---|---|
| B = | | x ↑ | **Bearbeite Formaloperand**<br>Die substituierten Bausteine werden unabhängig von Bedingungen (absolut) aufgerufen. | | |
| Formaloperanden | | | zulässige Aktualoperanden | Parameter Art | Parameter Typ |
| | | | DB, PB, SB, FB *1 | B | |

*1 Funktionsbausteine dürfen als Aktualoperanden keine Bausteinparameter enthalten.

**Beispiel:**

| Aufruf des FB | | Programm im FB 35 | | ausgeführtes Programm | |
|---|---|---|---|---|---|
| | :SPA FB 35 | :B | =D5 | :A | DB 5 |
| NAME | :BEARB. | :L | =DW2 | :L | DW 2 |
| D5 | :DB 5 | :B | =D6 | :A | DB 6 |
| DW2 | :DW 2 | :T | =DW1 | :T | DW1 |
| D6 | :DB6 | :T | =A4 | :T | AW 4 |
| DW1 | :DW1 | :B | =MOT5 | :SPA | FB 36 |
| A4 | :AW 4 | :BE | | :BE | |
| MOT5 | :FB36 | | | | |
| | :BE | | | | |

# Systemoperationen

Für die Systemoperationen gelten die gleichen Einschränkungen wie für ergänzende Operationen.

Sie können nur:

- in Funktionsbausteinen
- in der Darstellungsart AWL

programmiert werden.

Systemoperationen sollten nur von Anwendern mit sehr guten Systemkenntnissen eingesetzt werden, da sie einen Eingriff in die Systemdaten bedeuten.

Wenn Sie Systemoperationen programmieren wollen, müssen Sie bei der Voreinstellung am PG "Systembefehle JA" eingeben.

**Setzoperationen**

Wie bei den Bit-Operationen aus dem Bereich "ergänzende Operationen" können mit diesen Setzoperationen einzelne Bits verändert werden.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| SU | **x** | **x** | Bit unbedingt setzen<br>Im Bereich der Systemdaten wird ein bestimmtes Bit auf "1" gesetzt. |
| RU | **x** | **x** | Bit unbedingt rücksetzen<br>Im Bereich der Systemdaten wird ein bestimmtes Bit auf "0" gesetzt. |
| Kennzeichen<br>BS | | Parameter<br>0.0...255.15 | |

**Bearbeitung der Setzoperation:**

Die Ausführung der Operation ist unabhängig von VKE.

**Lade- und Transfer-operationen**

Mit diesen Operationen können Sie den gesamten Programmspeicher der CPU ansprechen. Sie werden vorwiegend zum Datenaustausch zwischen den Akkumulatoren und solchen Speicherplätzen verwendet, die nicht durch Operanden angesprochen werden können.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| LIR | | **x** | Lade Register indirekt |
| TIR | | **x** ↑ | Transferiere Register indirekt<br>Der Inhalt des angegebenen Registers wird zu einem Speicherplatz transferiert, dessen Adresse in AKKU 1 steht. |
| | | ↑ Parameter<br>0 (für AKKU 1), 2(für AKKU 2) | |
| LDI | **x** | | Lade Register indirekt.<br>Der angegebene AKKU wird mit dem Inhalt eines Speicherwortes geladen, dessen Adresse in AKKU 1 steht (Zugriff auf die zweiter Speicherbank, nur CPU 244) |
| TDI | **x** ↑ | | Transferiere Register indirekt.<br>Der Inhalt des angegebenen Registers wird zu einem Speicherplatz transferiert, dessen Adresse in AKKU 1 steht. (Zugriff auf die zweite Speicherbank, nur CPU 244). |
| Kennzeichen<br>A1 (für AKKU 1)<br>A2 (für AKKU 2) | | | |
| TNB | | **x** | Transferiere einen Datenblock (byteweise).<br>Ein Speicherbereich wird im Programmspeicher blockweise transferiert.<br>Endadresse Zielbereich: AKKU 1<br>Endadresse Quellbereich: AKKU 2 |
| T | **x** ↑ | **x** ↑ | Transferiere.<br>Ein Wort wird in den Systemdatenbereich transferiert. |
| Kennzeichen<br>BS | | ↑ Parameter<br>0...255 | |

**Sonderbefehle**

LIR 1, LIR 3, TIR 1 und TIR 3 ab (V1.07) Die Befehle sind analog zu den bekannten Befehlen LIR 0, LIR 2, TIR 0 und TIR 2 aufgebaut.

Die Befehle nutzen die AKKUS wie folgt:

| Befehl | AKKU 1 | AKKU 2 |
|---|---|---|
| LIR 1 | Inhalt des Speicherwortes (durch Quelladresse im DB adressiert). | Nummer des Datenwortes im DB, ab dem die Quelladresse zu finden ist. * |
| LIR 3 | Nummer des Datenwortes im DB, ab dem die Quelladresse zu finden ist. * | Inhalt des Speicherwortes (durch Quelladresse im DB adressiert). |
| TIR 1 | Inhalt der in das Speicherwort (durch Zieladresse im DB adressiert) transferiert wird. * | Nummer des Datenwortes im DB, ab dem die Zieladresse zu finden ist. |
| TIR 3 | Nummer des Datenwortes im DB, ab dem die Zieladresse zu finden ist. * | Inhalt der in das Speicherwort (durch Zieladresse im DB adressiert transferiert wird. * |

* Die AKKUs sind vor dem Befehlsaufruf entsprechend zu belegen

TNB und TNW für CPU 244 (ab V1.08)

Um den kompletten Speicher adressieren zu können werden zwei Worte zur Adressierung verwendet. Die Adressen werden in einem Datenbaustein (DB) hinterlegt. Der entsprechende DB ist vor dem Befehl zu öffnen (A DB y). Jede Adresse besteht aus zwei Datenworten, die aufeinanderfolgend im DB abzulegen sind.

DB y

| | |
|---|---|
| Highword der Adresse | Datenwort x |
| Lowword der Adresse | Datenwort x+1 |

die Befehle nutzen die AKKUs wie folgt:

| Befehl | AKKU 1 | AKKU 2 |
|---|---|---|
| TNB n<br>n = Anzahl der Bytes | Nummer des Datenwortes im DB, ab dem die Zieladresse zu finden ist. * | Nummer des Datenwortes im DB, ab dem die Quelladresse zu finden ist. * |
| TNW n<br>n = Anzahl der Bytes | Nummer des Datenwortes im DB, ab dem die Zieladresse zu finden ist. * | Nummer des Datenwortes im DB, ab dem die Quelladresse zu finden ist. * |

* Die AKKUs sind vor dem Befehlsaufruf entsprechend zu belegen.

Die Adressen im DB werden um die Anzahl der übertragenen Bytes verringert.

Der TNW Befehl akzeptiert nur gerade Adressen.

**Laden und Transferieren**

**von Registerinhalten**

Die beiden AKKUs sind als Register ansprechbar. Jedes Register ist 16 Bit breit. Da die beiden Operationen "LIR" und "TIR" die Daten wortweise übertragen, werden die Register in der CPU 24x paarweise angesprochen.

Die Ausführung der Operationen ist unabhängig vom VKE. Die Adresse des Speicherplatzes, der beim Datenaustausch angesprochen wird, entnimmt das Steuerwerk dem AKKU 1.

Bevor die Systemoperation bearbeitet wird, müssen Sie also dafür sorgen, dass die gewünschte Adresse im AKKU 1 hinterlegt ist.

| AWL | Erläuterung |
|---|---|
| .<br>. | |
| L KH F100 | Die Adresse F100h wird in den AKKU 1 geladen |
| LIR 0 | Die Information wird vom Speicherplatz mit der Adresse F100h in den AKKU 1 geladen |

**Beispiel:** Der Inhalt der Speicherzelle 1231h und 1232h auf der zweiten Speicherbank soll in AKKU 2 geladen werden.

Der Inhalt von Speicherzelle 1231h sei 45h;
Der Inhalt von Speicherzelle 1232h sei 67h.

| AWL | Erläuterung |
|---|---|
| . | |
| L KH 1231 | Die Adresse 1231h wird in den AKKU 1 geladen |
| LDI A2 | Nach dieser Operation steht in AKKU 2 der Inhalt der Speicherzellen 1231h und 1232h, also 4567h. |

**Beispiel:** In den Speicherzellen 1231h und 1232h der zweiten Speicherbank sollen die Werte 44h und 66h transferiert werden.

| AWL | Erläuterung |
|---|---|
| . | |
| L KH 4466 | Die Konstante 4466h wird in den AKKU 1 geladen |
| L KH 1231 | Nach dieser Operation steht in AKKU 1: 1231h und in AKKU 2: 4466h |
| TDI A2 | Nach der Transferoperation seht im Speicherzelle 1231h der Wert 44h und in Speicherzelle 1232h der Wert 66h. |

**Lade und Transfer**

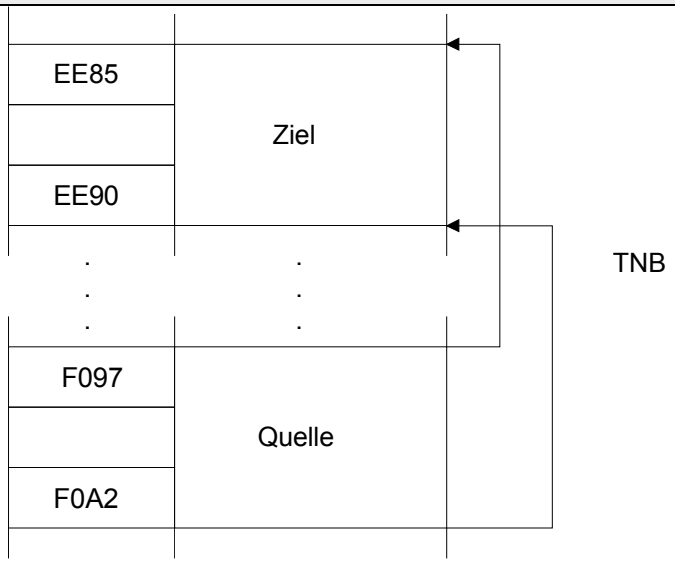
**Bearbeitung des Blocktransfers:**

Die Ausführung der Operation ist unabhängig vom VKE.

Der Parameter gibt die Länge des Datenblocks (in Bytes) an, der transferiert werden soll. Die Blocklänge kann höchstens 255 Byte betragen.

Die Adresse des Quellenfeldes wird dem AKKU 2 entnommen, die Adresse des Zielfeldes steht im AKKU 1.

Der Blocktransfer erfolgt dekrementierend, d.h. es müssen jeweils die oberen Adressen der Felder angegeben werden. Beim Transfer werden die Bytes im Zielfeld überschrieben!

| Beispiel | Darstellung |
|---|---|
| Ein Datenblock von 12 Bytes soll von der Adresse F0A2h zur Adresse EE90h transferiert werden. |  |



| AWL | Erläuterung |
|---|---|
| .<br>L KH F0A2 | Die Endadresse des Quellenfeldes wird in den AKKU 1 geladen. |
| .<br>L KH EE90 | Die Endadresse des Zielfeldes wird in den AKKU 1 geladen. Die Quellenadresse wird in den AKKU 2 geschoben. |
| .<br>TNB 12 | Der Datenblock wird ins Zielfeld transferiert. |

**Lade- und Transfer**

**Transferieren in den Systemdatenbereich**

**Beispiel:** Nach jedem Betriebsartenwechsel STOP→RUN soll die Zyklusüberwachungszeit auf 100 ms eingestellt werden. Diese Zeit kann als Vielfaches von 10 ms im Systemdatenwort 96 programmiert werden (nicht bei CPU 241). Der folgende Funktionsbaustein kann z. B. vom OB 21 aus aufgerufen werden:

| AWL | Erläuterung |
|---|---|
| .<br>FB 11 | Art und Nummer des Bausteins. |
| .<br>L KF 10 | AKKU 1 wird mit dem Faktor 10 geladen. |
| .<br>T BS 96<br>BE | Dieser Wert wird ins Systemdatenwort 96 übertragen. |

**Hinweis!**

Die Operationen TIR, TDI, T BS und TNB sind speicherverändernde Operationen, mit denen Sie Zugriffe auf den Anwenderspeicher und den Systemdatenbereich durchführen können, die nicht vom Betriebssystem überwacht werden. Eine unsachgemäße Verwendung der Operationen kann zur Programmveränderung und zum CPU-Absturz führen.

**Sprungoperation**

Innerhalb von Funktionsbausteinen kann ein Sprungziel durch eine Marke festgelegt werden. Bei dieser Sprungoperation können Sie die Sprungdistanz durch eine Festpunktzahl angeben. Die wichtigsten Eigenschaften sind in folgender Tabelle aufgeführt.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| SPR | | **x** | **Springe relativ**<br>Die lineare Programmbearbeitung wird unterbrochen und an der Stelle fortgesetzt, die durch die Sprungdistanz festgelegt ist. |
| | | Parameter<br>-32768...+32767 | |

**Bearbeitung der Sprungoperationen:**

Die Ausführung der Operation ist unabhängig vom VKE.

Die Sprungdistanz wird direkt durch den Parameter angegeben. So bedeutet z.B. der Parameter "2", dass nicht mit der nächsten, sondern erst mit der übernächsten 1-Wort-Anweisung weitergearbeitet wird.

Diese Markierung hat folgende Besonderheiten:

- Die Sprungdistanz wird nicht automatisch nachgeführt. Wird der übersprungene Programmteil verändert, so kann dadurch das Sprungziel verschoben werden.
- Das Sprungziel sollte im gleichen Netzwerk oder Baustein liegen wie die Sprunganweisung.

**Hinweis!**

Da Sie keinen Einfluss auf die absolute Lage der Bausteine im internen Anwenderspeicher haben, sollten Sie Sprungziele vermeiden, die außerhalb der Bausteingrenze liegen.

**Arithmetische Operation**

Die Operation erhöht den Inhalt des AKKU 1 um den angegebenen Wert. Dieser Wert wird als positive oder negative Dezimalzahl durch den Parameter dargestellt.

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| ADD | **x** ↑ | **x** ↑ | **Addiere eine Konstante**<br>Es können Byte- oder Wortkonstanten addiert werden |
| Kennzeichen | | Parameter | |
| | BF | -128...+127 | |
| | KF | -32768...+32767 | |

**Bearbeitung:**

Die Operation wird unabhängig vom VKE ausgeführt. Sie beeinflusst andererseits weder das VKE noch die Anzeigen.

Durch Eingeben negativer Parameter können auch Subtraktionen durchgeführt werden. Auch wenn das Ergebnis nicht durch 16Bit dargestellt werden kann, erfolgt kein Übertrag auf den AKKU 2, d.h. der Inhalt von AKKU 2 bleibt unverändert.

| Beispiel | AWL | Erläuterung |
|---|---|---|
| Die Hexakonstante 1020h soll um 33 erniedrigt und das Ergebnis im Merkerwort 28 abgelegt werden. Anschließend soll zum Ergebnis die Konstante 256 addiert und die Summe im Merkerwort 30 hinterlegt werden | L KH1020 | Die Konstante 1020h wird in den AKKU 1 geladen. |
| | ADD BF-33 | Zum AKKU-Inhalt wird die Konstante $-33_{10}$ addiert. |
| | T MW 28 | Der neue AKKU-Inhalt (0FFFh) wird in MW 28 abgelegt. |
| | ADD KF 256 | Zum letzten Ergebnis wird die Konstante $256_{10}$ addiert |
| | T MW 30 | Der neue AKKU-Inhalt (10FFh) wird in MW 30 abgelegt. |

**Sonstige Operationen**

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| BI | | | **Bearbeite Indirekt**<br>Über einen Formaloperanden wird eine Operation indiziert; bei der Ausführung der Operation wird der Bausteinparameter bearbeitet, dessen Nummer im AKKU 1 steht. |

**Bearbeitung**

Die Operation "BI" arbeitet wie die anderen Bearbeitungsoperationen. Im Gegensatz zu "B DW" oder "B MW" wird bei dieser Operation ein Formaloperand indiziert. Die Anweisung, die durch "BI" ausgeführt wird, bezieht sich auf den angegebenen Formaloperand. Dieser wird jedoch nicht durch seine Bezeichnung angegeben. Vor der "BI"-Anweisung müssen Sie den AKKU 1 mit der "Platznummer" des Formaloperanden in der Parameterliste laden.

| aufrufender Baustein | parametrierter FB | Erläuterung |
|---|---|---|
| SPA FB 2<br>NAME: BEARB<br>EIN 0 EW 10<br>EIN 1 EW 20<br>AUS AW 100 | NAME: BEARB<br>BEZ : EIN 0 EW<br>BEZ : EIN 1 EW<br>BEZ :AUS AW<br>.<br>.<br>. | |
| | L KF +2 | Der AKKU 1 wird mit der Konstanten "2"geladen. |
| | BI | Mit der nächsten Anweisung soll der Formaloperand bearbeitet werden, der an der zweiten Stelle der Parameterliste steht. |
| | T AW 80 | Der Inhalt von EW 20 wird zum AW 80 transferiert. |

**Sonstige Operationen**

Die Operationen "TAK" und "STS".

| Operation | Operand | | Bedeutung |
|---|---|---|---|
| TAK | | | **Tausche Akkumulatorinhalt**<br>Unabhängig vom VKE werden die Inhalte von AKKU 1 und AKKU 2 vertauscht. Das VKE und die Anzeigen werden nicht beeinflusst. |
| STS | | | **Stop sofort**<br>Unabhängig vom VKE wird die CPU in STOP gebracht. |

**Bearbeitung der Stopp-Operation:**

Bei der Ausführung der Operation "STS" geht die CPU sofort in STOP über, die Programmbearbeitung wird an dieser Stelle abgebrochen. Der STOP-Zustand kann nur manuell (Betriebsartenschalter) oder mit der PG-Funktion "AG-START" verlassen werden.

# Anzeigenbildung

Das Steuerwerk besitzt drei Anzeigen:

- ANZ 0,
- ANZ 1,
- OV Überlauf (Overflow).

Die Anzeigen werden von verschiedenen Operationen beeinflusst:

- Vergleichsoperationen,
- Rechenoperationen,
- Schiebeoperationen,
- und einigen Umwandlungsoperationen.

Die Belegung der Anzeigen stellt dann eine Bedingung für die verschiedenen Sprungoperationen dar.

Anzeigenbildung bei Vergleichsoperationen:

Die Ausführung der Vergleichsoperationen führt zum Setzen der Anzeigen ANZ 0 und ANZ 1. Die Überlaufanzeige wird nicht verändert. Die Vergleichsoperationen beeinflussen jedoch das Verknüpfungsergebnis. Bei erfüllter Aussage ist das VKE = 1. Deshalb kann auch die bedingte Sprungoperation "SPB" nach einer Vergleichsoperation eingesetzt werden.

| Inhalt von AKKU 2 gegenüber Inhalt von AKKU 1 | Anzeigen | | | mögliche Sprungoperationen |
|---|---|---|---|---|
| | ANZ 1 | ANZ 0 | O V | |
| gleich | 0 | 0 | | SPZ |
| kleiner | 0 | 1 | | SPN, SPM |
| größer | 1 | 0 | | SPN, SPP |

**Anzeigen**

**Anzeigen bei arithmetischen Operationen**

Die Ausführung der Rechenoperationen führt zum Setzen aller Anzeigen, je nach Ergebnis der Rechenoperation.

| Ergebnis nach Ausführung der Rechenoperation | Anzeigen | | | mögliche Sprungoperationen |
|---|---|---|---|---|
| | ANZ 1 | ANZ 0 | O V | |
| <-32768 | 1 | 0 | 1 | SPN, SPP, SPO |
| -32768 bis -1 | 0 | 1 | 0 | SPN, SPM |
| 0 | 0 | 0 | 0 | SPZ |
| +1 bis +32767 | 1 | 0 | 0 | SPN, SPP |
| >+32767 | 0 | 1 | 1 | SPN, SPM, SPO |
| (-)65536* | 0 | 0 | 1 | SPZ, SPO |

* Ergebnis der Rechnung:-32768 - 32768

**Anzeigenbildung bei wortweisen Verknüpfungen**

Die Digitalverknüpfungen führen zum Setzen der Anzeigen ANZ 0 und ANZ 1. Die Überlaufsanzeige wird nicht beeinflusst. Das Setzen der Anzeigen hängt vom Inhalt des AKKUs nach der Bearbeitung der Operation ab:

| Inhalt des AKKUs | Anzeigen | | | mögliche Sprungoperationen |
|---|---|---|---|---|
| | ANZ 1 | ANZ 0 | O V | |
| Null (KH = 0000) | 0 | 0 | | SPZ |
| nicht Null | 1 | 0 | | SPN, SPP |

**Anzeigen**

**Anzeigenbildung bei Schiebeoperationen**

Die Ausführung der Schiebeoperationen führt zum Setzen der Anzeigen ANZ 0 und ANZ 1. Die Überlaufanzeige wird nicht beeinflusst.

Das Setzen der Anzeigen hängt vom Zustand des zuletzt hinausgeschobenen Bits ab.

| Wert des zuletzt hinausgeschobenen Bits | Anzeigen | | | mögliche Sprungoperationen |
|---|---|---|---|---|
| | ANZ 1 | ANZ 0 | O V | |
| "0" | 0 | 0 | | SPZ |
| "1" | 1 | 0 | | SPN, SPP |

**Anzeigenbildung bei Umwandlungsoperationen**

Die Bildung des Zweierkomplements (KZW) führt zum Setzen aller Anzeigen. Die Belegung der Anzeigen richtet sich nach dem Ergebnis der Umwandlungsfunktion.

| Ergebnis nach Ausführung der Rechenoperation | Anzeigen | | | mögliche Sprungoperationen |
|---|---|---|---|---|
| | ANZ 1 | ANZ 0 | O V | |
| -32768* | 0 | 1 | 1 | SPN, SPM, SPO |
| -32767 bis -1 | 0 | 1 | 0 | SPN, SPM |
| 0 | 0 | 0 | 0 | SPZ |
| +1 bis +32767 | 1 | 0 | 0 | SPN, SPP |

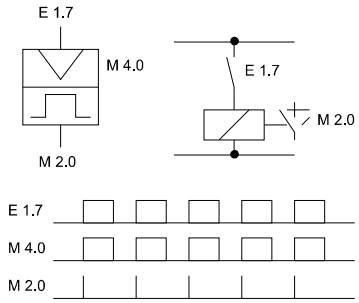
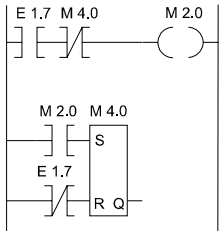
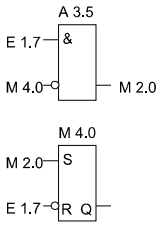
* Ergebnis der Umwandlung von KH = 8000

# Programmbeispiele

Im folgenden Abschnitt finden Sie einige Programmbeispiele, die Sie an einem PG mit Bildschirm (z.B. PG 740) in allen drei Darstellungsarten programmieren und testen können.

**Wischrelais** (Flankenauswertung)

*Nachbildung eines Wischrelais*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|  | U E 1.7<br>UN M 4.0<br>= M 2.0<br>U M 2.0<br>S M 4.0<br>UN E 1.7<br>R M 4.0 |  |  |

Bei jeder ansteigenden Flanke des Eingangs E 1.7 ist die UND-Verknüpfung (U E 1.7 und UN M 4.0) erfüllt und mit VKE = "1" werden die Merker M 4.0 ("Flankenmerker") und M 2.0 ("Impulsmerker") gesetzt.

Beim nächsten Bearbeitungszyklus ist die UND-Verknüpfung U E 1.7 und UN M 4.0 nicht erfüllt, da der Merker M 4.0 gesetzt worden ist.

Der Merker M 2.0 wird zurückgesetzt.

Der Merker M 2.0 führt also während eines einzigen Programmdurchlaufs Signalzustand "1".

Beim Ausschalten des Eingangs 1.7 wird der Merker 4.0 zurückgesetzt.

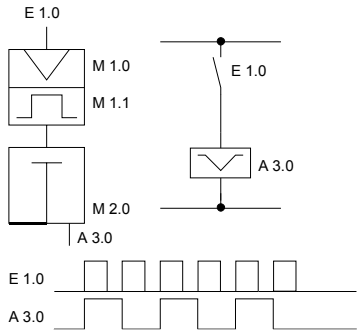
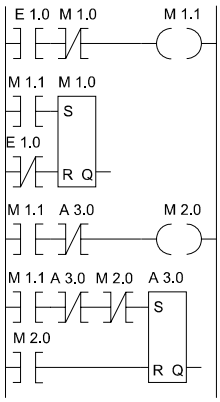
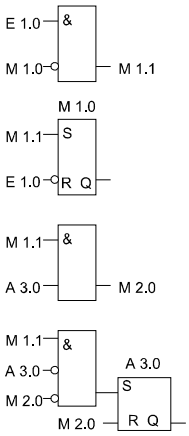
Die Auswertung der nächsten ansteigenden Flanke des Eingangs 1.7 wird somit vorbereitet.

**Binäruntersetzer (T-Kippglied)**

Dieser Abschnitt beschreibt, wie ein Binäruntersetzer programmiert wird.

**Beispiel:**

*Binärumsetzer (T-Kippglied)*

| Aufgabenstellung | Darstellung | | |
|---|---|---|---|
| | AWL | KOP | FUP |
|  | U E 1.0<br>UN M 1.0<br>= M 1.1<br>U M 1.1<br>S M 1.0<br>UN E 1.0<br>R M 1.0<br>U M 1.1<br>U A 3.0<br>= M 2.0<br>U M 1.1<br>UN A 3.0<br>UN M 2.0<br>S A 3.0<br>U M 2.0<br>R A 3.0 |  |  |



Der Binärumsetzer (Ausgang A 3.0) wechselt bei jedem Signalzustandswechsel von "0" nach "1" (ansteigende Flanke) des Eingangs E 1.0 seinen Zustand. Am Ausgang des Speichergliedes erscheint deshalb die halbe Eingangsfrequenz.

**Hinweis!**

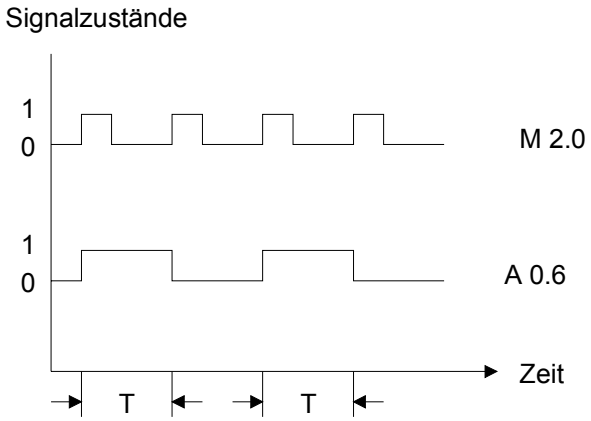
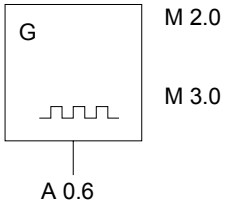
Die Ausgabe in FUP oder KOP ist nur möglich, wenn bei Programmierung in AWL die Segmentgrenzen "***" eingetragen werden.

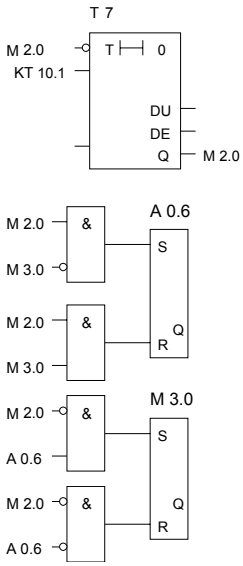
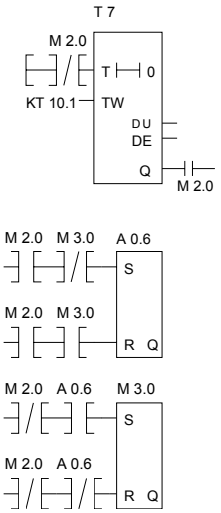
**Taktgeber (Taktgenerator)**

Im folgenden wird beschrieben, wie ein Taktgeber programmiert wird

Beispiel: Ein Taktgeber kann durch ein selbsttaktendes Zeitglied, dem ein T-Kippglied (Binäruntersetzer) nachgeschaltet ist, aufgebaut werden. Mit dem Merker 2.0 wird die Zeit 7 nach jedem Ablauf neu gestartet, d.h. der Merker 2.0 führt nach jedem Ablauf der Zeit für einen Zyklus den Signalzustand "1". Diese Impulse des Merkers 2.0 wirken auf das nachfolgende T-Kippglied, so dass am Ausgang 0.6 eine Impulsfolge mit dem Tastverhältnis 1:1 erscheint. Die Periodendauer dieser Impulsfolge ist doppelt so groß wie der Zeitwert des selbsttaktenden Zeitgliedes.

Taktgeber

| Zeitdiagramm | Stromlaufplan |
|---|---|
|   |   |

| AWL | FUP | KOP |
|---|---|---|
| UN M 2.0<br>L KT 10.1<br>SE T 7<br>NOP 0<br>NOP 0<br>NOP 0<br>U T 7<br>= M 2.0<br>***<br>U M 2.0<br>UN M 3.0<br>S A 0.6<br>U M 2.0<br>U M 3.0<br>R A 0.6<br>NOP 0<br>***<br>UN M 2.0<br>U A 0.6<br>S M 3.0<br>UN M 2.0<br>UN A 0.6<br>R M 3.0<br>NOP 0 |   |   |

**Verzögerungs-zeiten**

Im folgenden Beispiel wird gezeigt, wie sie mit einem Zeitglied Verzögerungszeiten programmieren können, um längere Wartezeiten zu realisieren.

| FB 23 AWL | Erläuterung |
|---|---|
| Blatt 1 | |
| Netzwerk 1 | |
| | |
| NAME :WARTE | Programmierte Verzögerung |
| :O M 0.0 | VKE "1" erzwingen |
| :ON M 0.0 | |
| : | |
| :L KT 100.0 | Eine Sekunde |
| :SE T 0 | Timer starten |
| SCHL :UN T 0 | Schleife |
| :SPA OB 31 | Zykluszeit nachtriggern, nicht in ANLAUF-OB |
| :SPB =SCHL | |
| :U T 0 | |
| :R T 0 | Timer zurücksetzen |
| :U T 0 | |
| :L KT 1.0 | Timer mit VKE "0" durchlaufen, |
| :SE T 0 | damit erneuter Anstoß möglich ist |
| :BE | |

**Achtung!**

Während der programmierten Wartezeit wird Ihr SPS-Programm nicht bearbeitet; d.h. die Steuerung reagiert nicht auf externe Ereignisse!

Die Zykluszeitüberwachung kann ansprechen (OB 31)!

Bei kleinen Zeiten (bis ca. 60 ms) können Sie den OB 160 nutzen.

**Beispiel:**

```
Verzögerungszeit von 30 ms programmieren:
L    KF +30000
SPA  OB 160
```

# Teil 10 Integrierte Bausteine

**Überblick**

Das Kapitel beginnt mit einer Übersicht aller integrierten Bausteine. Nachfolgend sind die Bausteine beschrieben. Hier finden Sie auch die Beschreibung der (Standard)-Hantierungsbausteine, wie sie bei Kommunikationsaufgaben eingesetzt werden.

Weiter finden Sie hier die integrierten OBs aufgeführt. Mit der Beschreibung des integrierten DB 1 und dessen Programmierung endet das Kapitel.

**Inhalt**


**Thema** **Seite**

**Teil 10 Integrierte Bausteine ..... 10-1**

Integrierte Funktionen ..... 10-2
FB 238 - COMPR (komprimieren) ..... 10-3
FB 239 - DELETE (löschen) ..... 10-4
FB 240 - COD:B4 (Codewandlung BCD/DUAL) ..... 10-5
FB 241 - COD:16 (Codewandler DUAL/BCD) ..... 10-6
FB 242 - MUL:16 (Multiplizierer) ..... 10-7
FB 243 - DIV:16 (Dividierer) ..... 10-8
FB 250 - RLG:AE (Analogwert lesen und normieren) ..... 10-9
FB 251 - RLG:AA (Analogwert ausgeben) ..... 10-12
Beispiel für eine Analogwertverarbeitung ..... 10-14
Integrierte Hantierungsbausteine ..... 10-17
Die Parameter der Hantierungsbausteine ..... 10-18
Parameterbeschreibung der Hantierungsbausteine ..... 10-19
Parametrierung von SSNR, A-NR, ANZW und BLGR ..... 10-21
Indirekte Parametrierung von Quell- und Zielangaben ..... 10-24
Tabelle über die möglichen QTYP/ZTYP - Parameter ..... 10-25
Aufbau des Anzeigenwortes ..... 10-27
Status und Fehleranzeige im Anzeigenwort ..... 10-28
Anzeigenwort ..... 10-29
Längen - Wort ..... 10-35
Aufbau des Parametrierfehler - Anzeigenbytes ..... 10-36
Einstellbare Blockgrößen ..... 10-37
FB 244 - SEND ..... 10-38
FB 245 - RECEIVE ..... 10-41
FB 246 - FETCH ..... 10-44
FB 247 - CONTROL ..... 10-45
FB 248 - RESET ..... 10-46
FB 249 - SYCHRON ..... 10-47
Integrierte Sonder-Organisationsbausteine ..... 10-48
OB 31 - Zykluszeittriggerung ..... 10-49
OB 160 - Variable Zeitschleife ..... 10-50
OB 251 - PID-Regelalgorithmus ..... 10-51
Integrierter Baustein DB 1 ..... 10-61

# Integrierte Funktionen

**Übersicht**

Integrierte FBs und OBs sind in Maschinensprache programmiert und laufen deshalb mit hoher Geschwindigkeit ab. Sie belegen keinen Platz im internen Programmspeicher.

Die integrierten Bausteine werden wie alle Bausteine, die im Steuerungsprogramm aufgerufen; sie können nur durch Prozessalarme unterbrochen werden.

In diesem Kapitel sind Bausteine, die im Steuerungsprogramm für spezielle Funktionen aufgerufen werden können. Nicht beschrieben sind solche Bausteine, die das Betriebssystem der CPU automatisch bei bestimmten Gegebenheiten (z.B. Programmier- und Gerätefehler) aufruft.

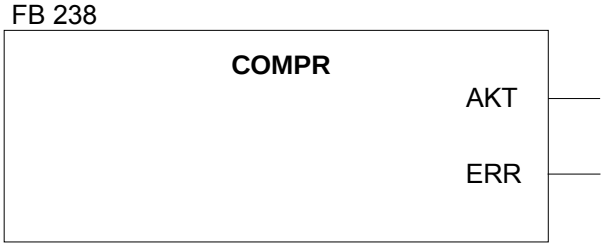
| Art | Nr. | Titel | Aufruflänge (in Worten) | Funktion |
|---|---|---|---|---|
| Integrierte Funktionen | | | | |
| FB | 238 | COMPR * | 4 | AG komprimieren |
| FB | 239 | DELETE | 5 | Baustein löschen |
| FB | 240 | COD:B4 | 5 | 4-Tetraden-BCD-Codewandler |
| FB | 241 | COD:16 | 6 | 16-bit-Festpunkt-Codewandler |
| FB | 242 | MUL:16 | 7 | 16-bit Dualmultiplizierer |
| FB | 243 | DIV:16 | 10 | 16-bit-Dualdividierer |
| FB<br>FB | 250<br>251 | RGL:AE<br>RLG:AA | 11<br>9 | Analogwert einlesen<br>Analogwert ausgeben |
| Integrierte Hantierungsbausteine | | | | |
| FB<br>FB<br>FB<br>FB<br>FB<br>FB | 244<br>245<br>246<br>247<br>248<br>249 | SEND **<br>RECEIVE **<br>FETCH<br>CONTROL<br>RESET<br>SYNCHRON | | Daten senden<br>Daten empfangen<br>Daten holen<br>Auftragsbearbeitung überwachen<br>Auftrag löschen<br>Schnittstelle einrichten |
| Integrierte Organisationsbausteine | | | | |
| OB | 31 | | | Zykluszeit neu starten |
| OB | 160 | | | Variable Zeitschleife |
| OB | 251 | | | PID-Regelalgorithmus |
| Integrierter Datenbaustein | | | | |
| DB | 1 | | | Interne Funktionen parametrieren |

* Laufzeit ohne den bausteinabhängigen Komprimiervorgang.

** Die Laufzeit ist von der Größe des zu übertragenden Datenblocks abhängig.

# FB 238 - COMPR (komprimieren)

Der integrierte FB 238 komprimiert bei seinem Aufruf den internen Programmspeicher.




**Parameter**

| | |
|---|---|
| AKT | Der FB 238 meldet mit dem Bit "AKT" zurück, ob die Funktion AG Komprimieren noch aktiv ist. |
| ERR | Durch das Bit "ERR" wird gemeldet, dass die Funktion nicht ausführbar ist. |

**Hinweis!**

Der FB COMPR wirkt wie die PG-Funktion "Komprimieren" , d.h. ist FB COMPR aktiv, so werden andere PG/OP-Funktionen zum Teil abgewiesen; z.B. STATUS oder Baustein Ein-/Ausgabe.

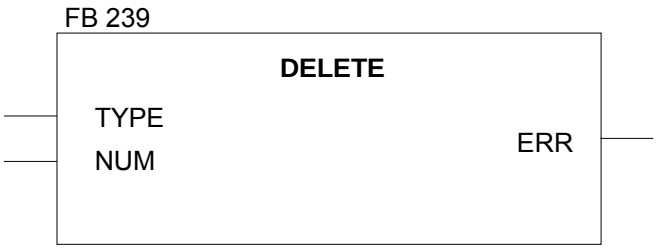
Auch das Erzeugen und Löschen eines DBs mit EDB führt in diesem Fall zu CPU-Stop (TRAF).

**Aufruf des FB 238**

```
                : U       E    0.0
                : UN      M    0.0
                : =       M    0.1
                : U       E    0.0
                : =       M    0.0
                :
                : U       M    0.1
                : SPB     FB   238
NAME            : COMPR
AKT             : M       1.0
ERR             : M       1.1
```

# FB 239 - DELETE (löschen)

Der integrierte FB 239 löscht einen Baustein.




**Parameter**

TYPE: Den TYP des zu löschenden Bausteins in einem Eingangs-, Merker-, oder Datenwort als ASCII-Zeichen (KC) hinterlegen. Zugelassen als Kennung sind OB, PB FB, SB und DB.

NUM: Baustein-Nr. in einem Eingangs- bzw. Merkerbyte hinterlegen.

ERR: Merker- oder Ausgangsbyte, in das Meldungen vom Betriebssystem eingetragen werden. Siehe folgende Tabelle

**Parameter ERR**

| Hexadezimaler Wert des Parameters ERR | Bedeutung |
|---|---|
| 00h | kein Fehler |
| F0h | Baustein nicht vorhanden |
| F1h | Falscher Bausteintyp in Parameter TYPE |
| F2h | Baustein vorhanden mit Kennung EPROM |
| F4h | Funktion gesperrt wegen anderer laufender Funktion (z.B. PG-Funktion) |

**Aufruf FB 239**

**AWL**

```
        : SPB     FB   239
NAME    : DELETE
TYPE    : MW      5        ASCII-codierte Bausteinbelegung
                           (z.B. PB für Programmbaustein)
NUM     : MB      7        : Bausteinnummer (z.B. KF+7)
ERR     : MB      8        MB 8 wird erst nach Aufruf dieses
                           FBs beschrieben
```

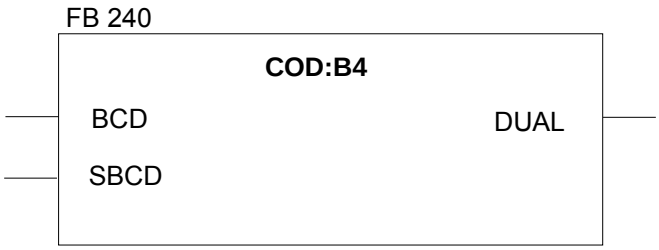
**Hinweis!**

Wenn DELETE gerade aktiv ist, dann können PG/OP-Funktionen abgewiesen werden.

# FB 240 - COD:B4 (Codewandlung BCD/DUAL)

Mit diesem Funktionsbaustein lässt sich eine BCD-Zahl (4 Tetraden) mit Vorzeichen in eine Festpunkt-Dualzahl (16 Bits ) umwandeln.

2 Tetraden-Zahlen müssen vor der Umwandlung in eine 4 Tetraden-Zahl transferiert, das heißt mit "0" aufgefüllt werden.



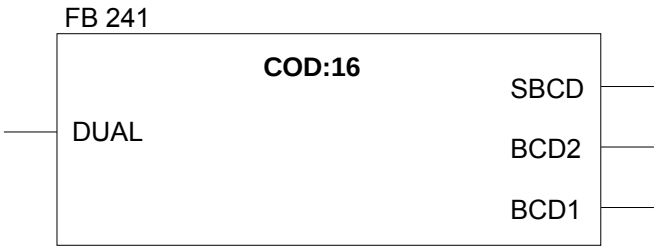

**Parameter**

| | |
|---|---|
| BCD: | BCD-Zahl als Wort -9999...+9999 |
| SBCD: | Vorzeichen als Bit der BCD-Zahl<br>"0" für +<br>"1" für - |
| DUAL: | Dualzahl als Wort 16 Bits |

# FB 241 - COD:16 (Codewandler DUAL/BCD)

Mit diesem Funktionsbaustein lässt sich eine Festpunkt-Dualzahl (16 Bits) in eine BCD-Zahl mit zusätzlicher Berücksichtigung des Vorzeichens umwandeln.

8 Bit-Dualzahlen müssen vor der Umwandlung in ein 16 Bit-Wort transferiert werden.



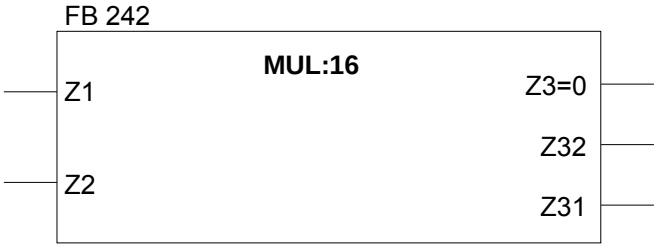

**Parameter**

| | |
|---|---|
| DUAL: | Dualzahl als Wort 16 Bits -32768...+32767 |
| SBCD: | Vorzeichen als Bit der BCD-Zahl<br>"1" für -<br>"0" für + |
| BCD1: | BCD-Zahl als Wort 0...3 Tetrade |
| BCD2: | BCD-Zahl als Byte 4. und 5. Tetrade |

# FB 242 - MUL:16 (Multiplizierer)

Mit diesem Funktionsbaustein lassen sich zwei Festpunkt-Dualzahlen (16 Bits) miteinander multiplizieren. Das Produkt wird durch zwei Festpunkt-Dualzahlen (je 16 Bits) dargestellt. Zusätzlich wird eine Abfrage des Ergebnisses auf Null durchgeführt.

8 Bit-Zahlen müssen vor der Multiplikation in 16 Bit-Wörter transferiert werden.




**Parameter**

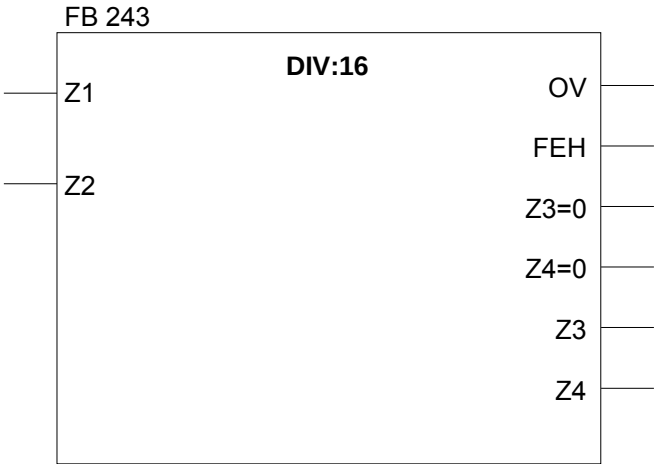
| | |
|---|---|
| Z1: | Multiplikator als Wort -32768...+32767 |
| Z2: | Multiplikant als Wort -32768...+32767 |
| Z3=0: | Abfrage auf 0; "1", falls das Produkt Null ist |
| Z32: | Produkt High-Wort - 16 Bits |
| Z31: | Produkt Low-Wort - 16 Bits |

# FB 243 - DIV:16 (Dividierer)

Mit diesem Funktionsbaustein lassen sich zwei Festpunkt-Dualzahlen (16 Bits) dividieren. Das Ergebnis (Quotient und Rest) wird durch zwei Festpunkt-Dualzahlen (je 16 Bits dargestellt.

Zusätzlich wird eine Abfrage des Divisors und des Ergebnisses auf Null durchgeführt.

8 Bit-Zahlen müssen vor der Division in 16 Bit-Wörter transferiert werden.




**Parameter**

| | |
|---|---|
| Z1: | Dividend als Wort -32768...+32767 |
| Z2: | Divisor als Wort -32768...+32767 |
| OV | Überlaufanzeige als Bit; "1" bei Überlauf |
| FEH | "1" bei Division durch Null |
| Z3=0: | Abfrage auf Null; "0", Quotient ist Null |
| Z4=0: | Abfrage auf Null; "0", Rest ist Null |
| Z3 | Quotient als Wort 16 Bits |
| Z4 | Rest als Wort 16 Bits |

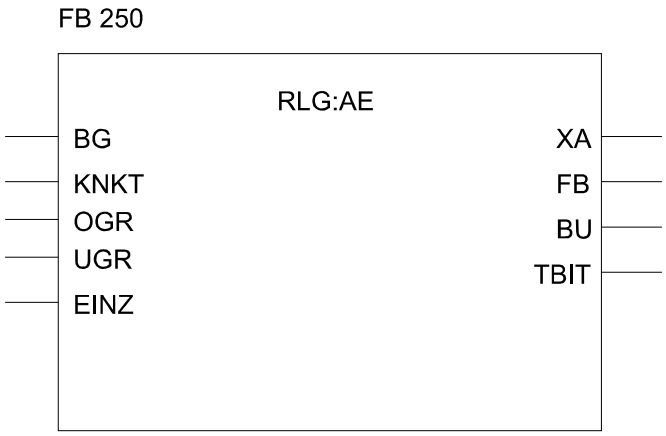
# FB 250 - RLG:AE (Analogwert lesen und normieren)

**Einleitung**

Dieser Funktionsbaustein liest einen Analogwert einer Analog-Eingabebaugruppe und liefert am Ausgang einen Wert XA in einem von Ihnen festgelegten (normierten) Bereich.

Mit den Parametern "Obergrenze (OGR)" und "Untergrenze (UGR)" legen Sie den gewünschten Bereich fest.

Die Art der Analogwertdarstellung der Baugruppe (Kanaltyp) muss im Parameter KNKT angegeben werden. Überschreitet der Analogwert den Nennbereich, wird der Parameter BU gesetzt.



**Hinweis**

Die Parametrierung der Analogmodule muss mit dem Kanaltyp übereinstimmen! Die Angabe wird nicht überprüft.

**Aufruf und Parameter**

| Parameter | Bedeutung | Art | Typ | Belegung | AWL |
|---|---|---|---|---|---|
| BG | Baugruppen-adresse | D | KF | 128...224 | : SPA FB 250<br>NAME: RLG:AE |
| KNKT | KN=Kanal-nummer<br>KT=Kanaltyp | D | KY | KY=x,y<br>x=0...15<br>y=3...6<br>3:Betragsdarst. (4...20mA)<br>4:unipolare Darstellung<br>5:Betragszahl bipolar. | BG :<br>KNKT:<br>OGR :<br>UGR :<br>EINZ:<br>XA :<br>FB :<br>BU :<br>TBIT: |
| OGR | Obergr. des Ausgangs-wertes | D | KF | -32768...<br>+32767 | |
| UGR | Untergr. des Ausgangs-wertes | D | KF | -32768...<br>+32767 | |
| EINZ | Einzelab-tastung | E | BI | irrelevant | |
| XA | Ausgangswert | A | BI | Normierter Analogwert ist "0" bei Drahtbruch. | |
| FB | Fehlerbit | A | BI | Ist "1" bei ungültiger Kanal- oder Steckplatznr., bei ungültigem Kanal-typ und QVZ der Baugruppe. | |
| BU | Bereichsüber-schreitung | A | BI | Ist "1" bei Über-schreitung des Nennbereichs. | |
| TBIT | Tätigkeitsbit des FBs | A | BI | Bei Signalzustand "1" führt der FB gerade eine Einzelabtastung durch. | |

KF = fester Wert

KY = Wort

**Normierungs-schema**

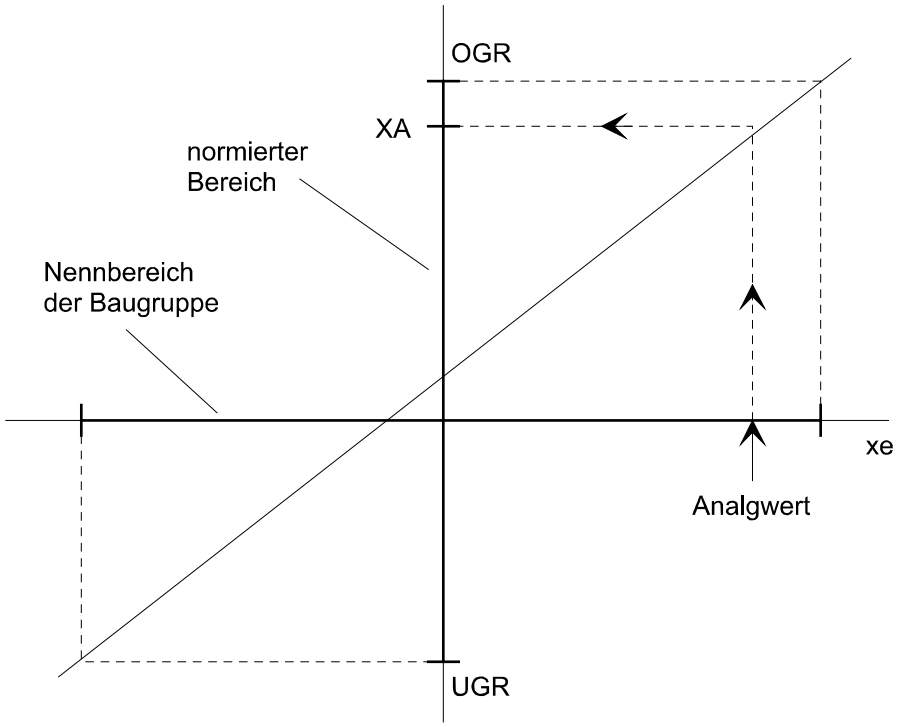
Der Funktionsbaustein FB 250 rechnet den gelesenen Wert linear auf die angegebene obere und untere Grenze (OGR und UGR) um, und zwar nach folgenden Formeln:

für Kanaltyp 3 (Betragszahl 4 bis 20mA):

für Kanaltyp 4 (unipolare Darstellung):

Kanaltyp 5 und 6 (bipolare Darstellung):

hierbei bedeutet : XA vom FB ausgegebener Wert
xe von der Baugruppe gelesener Analogwert



# FB 251 - RLG:AA (Analogwert ausgeben)

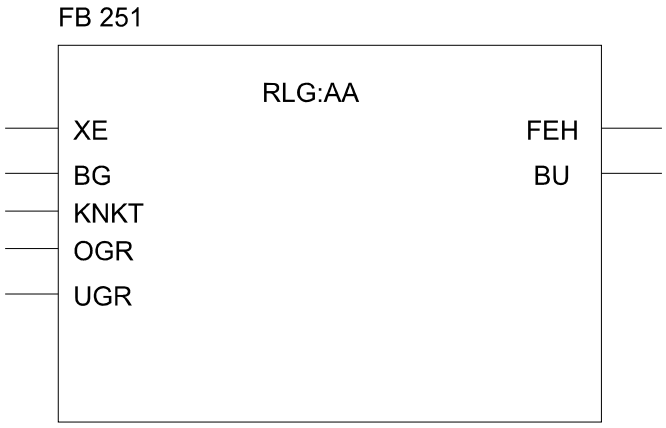
**Normierung**

Mit diesem Funktionsbaustein lassen sich Analogwerte an Analog-Ausgabebaugruppen ausgeben. Die Art der Analogwertdarstellung der Baugruppe (Kanaltyp) muss im Parameter KNKT angegeben werden. Dabei werden Werte aus dem Bereich zwischen den Parametern "Untergrenze (UGR)" und "Obergrenze (OGR)" auf den Nennbereich der jeweiligen Baugruppe umgerechnet, und zwar nach folgenden Formeln:

für Kanaltyp 0 (unipolare Darstellung):

für Kanaltyp 1 (bipolare Darstellung):

dabei bedeutet: XE am Funktionsbaustein angegebener Digitalwert
xa zur Baugruppe geschriebener Wert.



**Aufruf und Parametrierung**

| Parameter | Bedeutung | Art | Typ | Belegung | AWL |
|---|---|---|---|---|---|
| XE | anzugebender Analogwert | E | W | Eingangswert (Festpunkt) im Bereich UGR...OGR | |
| BG | Baugruppen-adresse | D | KF | 128...240 | : SPA FB 250<br>NAME: RLG:AE |
| KNKT | KN=Kanal-nummer<br>KT=Kanaltyp | D | KY | KY=x,y<br>x=0...7<br>y=0;1;2<br>0: unipolare Darstellung<br>1:Festpunktzahl bipolar<br>2: Betragsdarst. (4...20mA bzw.1,5V ) | BG :<br>KNKT:<br>OGR :<br>UGR :<br>EINZ:<br>XA :<br>FB :<br>BU :<br>TBIT: |
| OGR | Obergr. des Ausgangs-wertes | D | KF | -32768...<br>+32767 | |
| UGR | Untergr. des Ausgangs-wertes | D | KF | -32768...<br>+32767 | |
| FEH | Fehler bei der Grenzwert-vorgabe | A | BI | Ist "1",wenn UGR=OGR, bei ungültiger Kanal- oder Steckplatznr. bei ungültigem Kanaltyp und bei QVZ der Baugruppe. | |
| BU | anzugebender Analogwert überschreitet UGR oder OGR | A | BI | Bei "1" liegt XE außerhalb (UGR;OGR)<br>XE nimmt den Grenzwert an. | |

# Beispiel für eine Analogwertverarbeitung

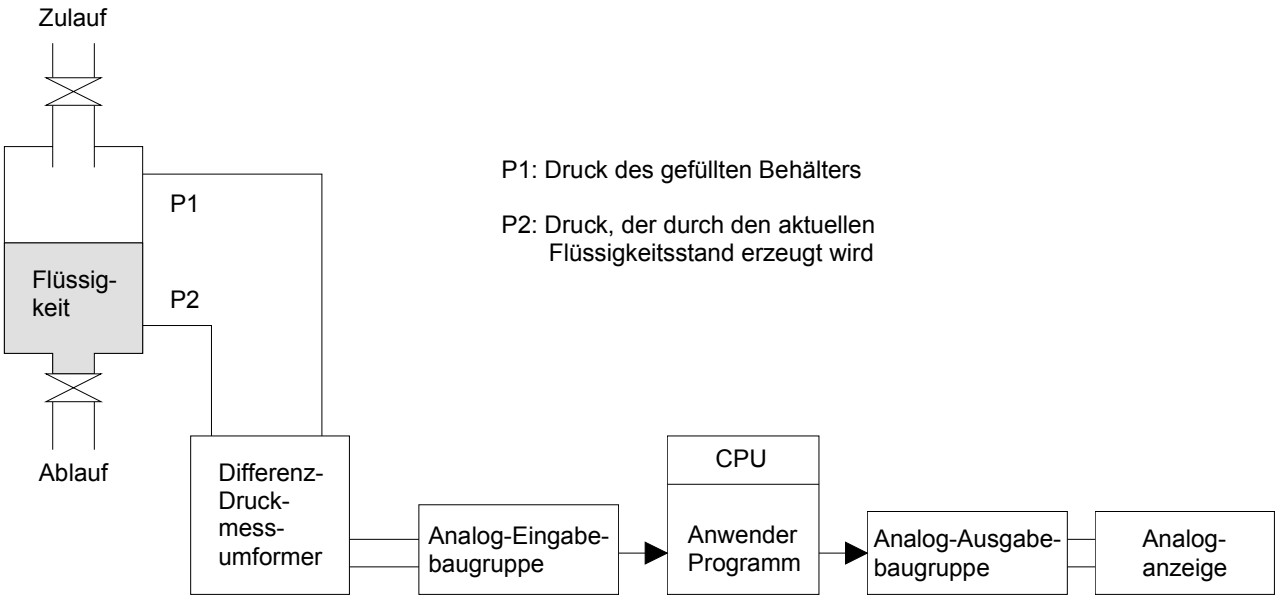
**Aufgabenstellung**

In einem geschlossenen Behälter befindet sich eine Flüssigkeit. Die aktuelle Niveauhöhe soll auf einem Anzeigeinstrument jederzeit abgelesen werden können.

Außerdem soll bei Erreichen eines vorgegebenen Grenzwertes eine Meldung ausgegeben werden.

**Beschreibung**

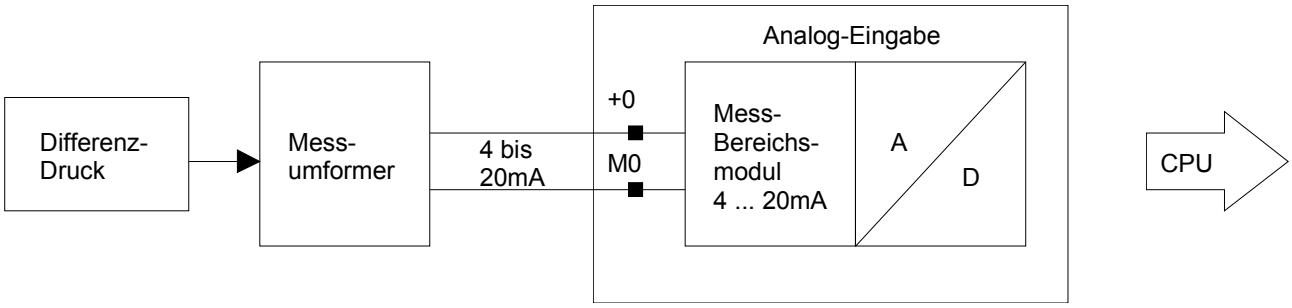
- Die Füllstandshöhe (zwischen 0 und 10m) wird von einem Messumformer 4 - 20mA an eine Analogeingabebaugruppe weitergegeben.
- Die Analogeingabebaugruppe wandelt die analogen Stromwerte in digitale Einheiten (0 - 2048 Einheiten) um, die vom Anwenderprogramm der CPU 24x weiterbearbeitet werden können.
- Das Anwenderprogramm prüft die eingelesenen Werte auf einen Grenzwert (max. zulässige Füllstandshöhe), gibt gegebenenfalls eine Meldung aus und übermittelt diese Werte an eine Analogausgabebaugruppe.
- Die Analogausgabebaugruppe setzt die Werte wieder in Spannungen (0 - 10V) um.
- Die Analoganzeige reagiert auf diese Spannungen mit einem der Füllstandshöhe proportionalen Zeigerausschlag.



**Inbetriebnahme**

**Analogeingabebaugruppe:**

- Messumformer direkt am Frontstecker der Eingabebaugruppe anschließen (Anschlusspunkte: +0, M0).
  Der Messumformer liefert Werte zwischen 4 und 20mA, wobei 4mA dem Stand 0,00m und 20mA dem Maximalstand 10,00m entsprechen.
- Messbereichsmodul 4 - 20mA in die Eingabebaugruppe stecken.
  Am Ausgang des internen A/D-Wandlers der Analog-Eingabebaugruppe liegt dann ein digitaler Wert zwischen 0 und 2048 Einheiten, der vom Anwenderprogramm verarbeitet wird. (siehe Abbildung)

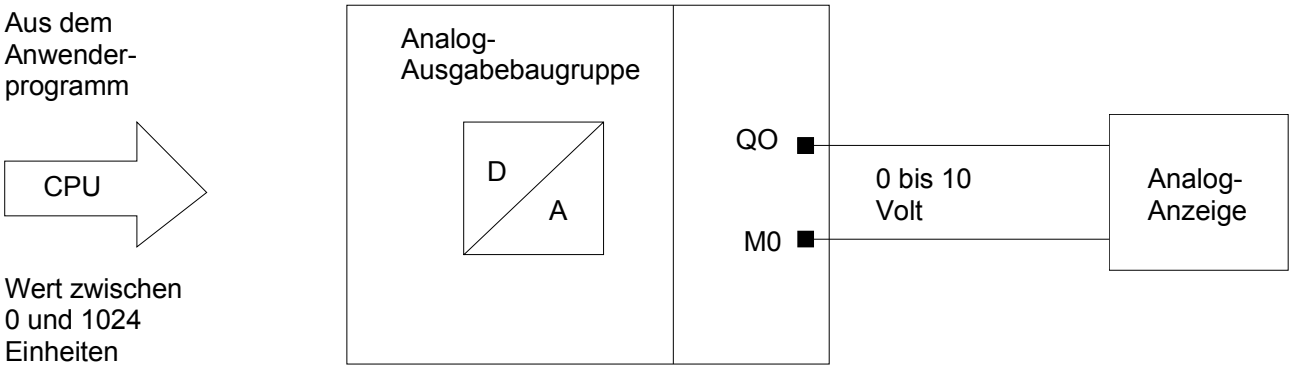



**Analog-Ausgabebaugruppe:**

- Anzeigeninstrument direkt am Frontstecker der Baugruppe anschließen

(Anschlusspunkte: QO, M0).

Von der Analog-Ausgabebaugruppe wird eine Spannung zwischen 0 und 10V an das Anzeigeinstrument ausgegeben, so dass am Instrument die Füllstandshöhe analog abgelesen werden kann (siehe Abbildung).



**Programmstruktur**

- Funktionsbaustein FB 250 "Analogwert einlesen" aufrufen und parametrieren (Umwandlung des Wertes in einem Bereich zwischen 0 und 1000cm [XA-Parameter]).
- Grenzwert bilden (PB 9).

Ein Überschreiten des Flüssigkeitsniveaus von 900cm verursacht eine Meldung (M12.6).

- Funktionsbaustein FB 251 "Analogwert ausgeben" aufrufen und parametrieren (Umwandlung des zwischen 0 und 1000 cm liegenden Wertes [XE-Parameter] in einen Wert zwischen 0 und 1024 Einheiten für die Analog-Ausgabebaugruppe).

**Beispiel**

PB 1:

```
          :SPA FB 250
NAME      :RLG:AE
BG        :KF     +128   Baugruppen-Anfangsadresse : 128
                         (bei fester Steckplatzadressierung
                         Steckplatz 0)
KNKT      :KY 0,4        Kanalnummer:0;unipolare Darstellung:4
OGR       :KF +1000      Physikalischer Messbereich:
UGR       :KF +0         0<XA<1000 cm
EINZ      :M 12.0        irrelevant:
                         (Einst. im Beispiel: Zykl. Bearb.)
XA        :MW 10         im MW 10: XA-Wert 0<XA<1000cm
FB        :M 12.1        relevant nur, wenn Drahtbruch
                         eingestellt ist
BU        :M 12.2        wenn Höhe > 1000cm, BU = 1.
TBIT      :M 12.3        nur bei Einzelabtastung relevant.
          :
          :SPA PB 9      Grenzwert bilden
          :
          :SPA FB 251    Analogwert ausgeben
NAME      :RLG:AA
XE        :MW 10         XA (FB 250) = XE (FB 251)
BG        :KF +160       Baugruppen-Anfangs-Adresse: 160
                         (bei fester Steckplatzadressierung
                         Steckplatz 1)
KNKT      :KY 0,0        Kanalnummer: 0;
                         unipolare Darstellung: 0
OGR       :KF +1000      Physikalischer Messbereich:
UGR       :KF +0         0<XA<1000cm
FEH       :M 12.4        wenn UGR = OGR, FEH = 1
BU        :M 12.5        wenn XA>UGR oder XA>OGR, BU = 1.
          :BE
```

PB 9:

```
          :L KF 900      Niveau Maximalwert
          :l MW 10       Messwert
          :<=F           Vergleiche, ob Messwert > 900 Ist
          :=M 12.6       wenn ja, M12.6 = Reaktion noch im
                         gleichen Programmzyklus auslesen.
          :BE
```

# Integrierte Hantierungsbausteine

**Was sind Hantierungsbausteine?**

Durch die mitgelieferten Hantierungsbausteine, wird der Einsatz von Kommunikationsprozessoren und IPs im System 200V ermöglicht.

Dadurch wird die Leistungsfähigkeit erheblich gesteigert.

Die Hantierungsbausteine steuern den gesamten Datenaustausch zwischen der CPU und den CPs, IPs.

Vorteile der Hantierungsbausteine:

- wenig Anwenderprogrammspeicherplatz geht verloren
- kurze Laufzeiten der Bausteine.

Die Hantierungsbausteine im System 200V benötigen keine:

- Merkerbereiche
- Zeitbereiche
- Zählerbereiche.

**Übersicht**

Folgende Hantierungsbausteine stehen zur Verfügung:

| Art | Nr. | Titel | Aufruflänge (in Wörtern | Funktion |
|---|---|---|---|---|
| FB | 244 | SEND * | | Daten senden |
| FB | 245 | RECEIVE * | | Daten empfangen |
| FB | 246 | FETCH | | Daten holen |
| FB | 247 | CONTROL | | Auftragsbearbeitung überwachen |
| FB | 248 | RESET | | Auftrag löschen |
| FB | 249 | SYNCHRON | | Schnittstelle einrichten |

* Die Laufzeit ist von der Größe des zu übertragenden Datenblocks abhängig.

# Die Parameter der Hantierungsbausteine

Alle nachfolgend behandelten Hantierungsbausteine haben eine einheitliche Schnittstelle zum Anwenderprogramm (siehe auch Parameterbeschreibung).

Die benutzten Parameter:

| | |
|---|---|
| SSNR | :Schnittstellennummer |
| A-NR | :Auftragsnummer |
| ANZW | :Anzeigenwort (Doppelwort) |
| QTYP/ZTYP* | :Typ der Datenquelle bzw. des Datenziel |
| DBNR* | :Datenbausteinnummer |
| QANF/ZANF* | :Relative Anfangsadresse innerhalb des Typs |
| QLAE/ZLAE* | :Anzahl der Quell-/Zieldaten |
| PAFE | :Parametrierungsfehler |
| BLGR | :Blockgröße |

*) Die bei einem bestimmten Aufruf nicht benötigten Parameter, (z.B bei "ALL-Funktion), können bei der Parametrierung des Bausteins mit "CR" übersprungen werden.

**direkte/indirekte Parametrierung**

Ein Hantierungsbaustein kann direkt oder indirekt parametriert werden. Nur der Parameter "PAFE" muss immer direkt angegeben werden.

Bei der direkten Parametrierung verarbeitet der Hantierungsbaustein die beim Bausteinaufruf angegebenen Parameter unmittelbar.

Bei der indirekten Parametrierung werden dem Hantierungsbaustein per Bausteinparameter Zeiger, die auf Parameterfelder (Datenbausteine bzw. Datenworte) zeigen, übergeben.

# Parameterbeschreibung der Hantierungsbausteine

**SSNR** Schnittstellen-Nummer

Nummer der logischen Schnittstelle (Kacheladresse) auf die sich der betreffende Auftrag bezieht.

Parameterart : Datum Byte (Format KY)

Sinnvoller Bereich : 0,0...0,255 (bei direkter Parametrierung) *

*) Ist das linke Byte >0 handelt es sich um eine indirekte Parametrierung von SSNR, A-NR; ANZW bzw. BLGR.

Das rechte Byte stellt in diesem Fall den Zeiger auf dem Parameterfeld dar (siehe Beispiele: Indirekte Parametrierung).

**A-NR** Auftragsnummer

Angesprochene Auftragsnummer für die logische Schnittstelle

Parameterart : Datum Byte (Format KY)

Sinnvoller Bereich : 0,0...0,223

Hierbei hat die Auftragsnummer 0 die Spezialfunktion "ALL", die beim FETCH Baustein nicht zugelassen ist

(siehe auch Beispiel ALL-Funktion).

**ANZW** Anzeigenwort (Doppelwort)

Adresse des Anzeigendoppelwortes im Anwenderspeicher, in dem die Abarbeitung des unter A-NR angegebenen Auftrages angezeigt wird.

Parameterart : Adresse Wort

Erlaubter Bereich : DW oder MW belegt wird jeweils DW und DW+1 oder MW und MW+2

Die Angabe DW bezieht sich auf den vor dem Aufruf aufgeschlagenen Datenbaustein.

Aufbau des ANZW: siehe Aufbau des Anzeigenwortes.

**QTYP/ZTYP** Typ der Datenquelle oder des Datenziels

An diesem Parameter kann mittels ASCII-Zeichen der Typ, der Datenquelle (SEND) oder des Datenziels (RECEIVE und FETCH) angegeben werden.

Parameterart :Datum Zeichen (Format KC)

Sinnvoller Bereich :DB, MB, AB, EB, ZB, TB, AS, NN, XX, RW

**DBNR** Datenbausteinnummer bei TYP XX, RW, DB

Wurde bei QTYP/ZTYP die Kennung XX, RW oder DB angegeben, muss bei diesem Parameter die Nummer des gewünschten Datenbausteins mitgeteilt werden.

Parameterart : Datum Byte (Formt KY)

Sinnvoller Bereich : 0,1...0,255

**QANF/ZANF** Anfangsadresse des Quell-/Ziel-Datenblocks

Bei Typkennung XX und RW (indirekte Parametrierung) wird hier die DW-Nummer angegeben, ab der die Parameter liegen. Ansonsten bezieht sich dieser Parameter absolut auf den angegebenen Bereich.

Parameterart : Datum Festpunkt (Format KF)

Möglicher Bereich : 0...+32767 bzw. bis -32768 bei "AS"

**QLAE/ZLAE** Länge des Quell-/Ziel-Datenblocks

Die Angabe ist je nach Quell-/Zieltyp in Bytes oder in Worten.

Parameterart : Datum Festpunktzahl (Format KF)

Möglicher Bereich : 0...+32767 (-1)*

*) Die "Jokerlänge" (QLAE, ZLAE (-1)) bedeutet beim RECEIVE, dass so viele Daten übernommen werden, wie der Sender liefert oder bis eigene Bereichsgrenze erreicht wurde.

Beim SEND, dass so viele Daten übergeben werden wie der Bereich zulässt.

**BLGR** Blockgröße

Bei Neustart wird mit Hilfe des Hantierungsbausteins "SYNCHRON" die Blockgröße (Größe der Datenblöcke) zwischen den Stationen ausgehandelt.

Hierbei bedeutet große Bockgröße = hoher Datendurchsatz aber auch lange Laufzeit und damit hohe Zykluszeitbelastung.

Kleine Blockgröße = kleiner Datendurchsatz aber auch kleine Laufzeiten der Bausteine.

Parameterart : Datum Byte (Format KY)

Möglicher Bereich : 0,0 ...0,255 (siehe Einstellbare Blockgrößen)

**PAFE** Fehleranzeige bei Parametrierungsfehler

Das hier angegebene "BYTE" (Ausgang, Merker) wird gesetzt, wenn der Baustein einen "Parametrierungsfehler" erkennt z.B. Schnittstelle (Anschaltung) nicht vorhanden oder unzulässige Parametrierung von QTYP/ZTYP; QANF/ZANF;QALAE/ZLAE erfolgte.

Parameterart : Ausgang, Byte

Sinnvoller Bereich : AB 0... AB 63 bei CPU 241; AB 0 ... AB127 bei CPU 242

: MB 0...MB 255

# Parametrierung von SSNR, A-NR, ANZW und BLGR

Das HIGH-Byte des Parameters SSNR dient als Umschaltkriterium für die direkte oder indirekte Parametrierung.

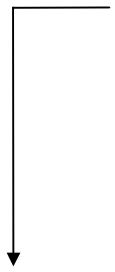
- HIGH-Byte von SSNR = 0 Bedeutet direkte Parametrierung
  SSNR, A-NR, ANZW oder BLGR sind direkt am Baustein vorgegeben.
- HIGH-Byte von SSNR ><0 Bedeutet indirekte Parametrierung
  SSNR, A-NR und ANZW/BLGR sind in dem aufgeschlagenen DB
  ab dem im LOW-Byte von SSNR angegebenen Datenwort abgelegt.

SSNR und A-NR haben in beiden Parametrierungsarten das gleiche Datenformat (KY). Beim Anzeigenwort ANZW unterscheiden sich die Darstellungsformate. Während bei der direkten Parametrierung die Adresse des Anzeigenwortes direkt (z.B. MW 100) angegeben wird, muss bei der indirekten Parametrierung eine zusätzliche Angabe über den Bereich des Anzeigenwortes erfolgen. Die Angabe dieses Bereiches steht in den, dem Anzeigenwort vorangestelltem Datenwort in ASCII-Code verschlüsselt.

Hierbei steht: MW für Anzeigenwort im Merkerbereich,
DB für Anzeigenwort im Datenbaustein.

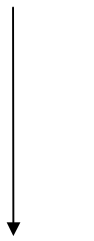
In dem darauf folgenden Datenwort des Parameterbereiches im DB steht im Datenformat KY die ANZW-Adresse, bei DB zusätzlich die Bausteinnummer (im ersten Byte des KY Formats).

```
Beispiele:
Direkte Parametrierung von SSNR, A-NR und ANZW
(ANZW = MW)

          SPA      FB245
          NAME    :RECEIVE
          SSNR    :KY 0,3      ;Parameter SSNR =3
          A-NR    :KY 0,100    ;Parameter A-NR = Auftrag NR
                                100
          ANZW    :MW 240      ;Parameter ANZW = MW 240
    .
    .
```



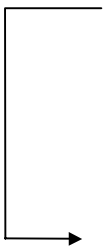
```
(ANZW = DW)

         A       DB     X
         .
         .
         SPA      FB247
         NAME    :CONTROL
         SSNR    :KY 0,3            ;Parameter SSNR = 3
         A-NR    :KY 0,100          ;Parameter A-NR = NR 100
         ANZW    :DW 40             ;Parameter ANZW =DW 40 im
                                     DB X
            .
            .



         DB      X

         DW 40
         DW 41
```

Indirekte Parametrierung von SSNR, A-NR und ANZW (Typ MW)



```
         A       DB X          ;Aufschlagen des Datenbausteins
                                "X"
         SPA     FB244
         NAME    :SEND
         SSNR    :KY 255,1     Kennung indirekte
                               Parametrierung
                               ;Wird als Zeiger interpretiert
         A-NR    :KY 0,0       ;Irrelevant
         ANZW    :MW 0         ;Irrelevant
            .
            .
         DB      X

         DW 1    KY 0,1        ;Parameter SSNR (=1)
         DW 2    KY 0,31       ;Parameter A-NR (31)
         DW 3    KC MW         Parameterart für ANZW (=MW)
         DW 4    KY 0,200      Adresse des Parameters ANZW
                               (MW 200 und MW 202)
```

Indirekte Parametrierung von SSNR, A-NR und ANZW (Typ DW)



```
   .
   .
A      DB X              ;Aufschlagen des
                          Datenbausteins "X"
SPA    FB244
NAME   :SEND
SSNR   :KY 255,1         ;Kennung indirekte
                          Parametrierung
                         ;Wird als Zeiger
                          interpretiert
A-NR   :KY 0,0           ;Irrelevant
ANZW   :MW 0             ;Irrelevant
   .
   .
DB     X

DW 1   KY 0,1            ;Parameter SSNR (=1)
DW 2   KY 0,31           ;Parameter A-NR (=31)
DW 3   KC DB             ;Parameterart für ANZW
                          (=DW)
DW 4   KY 222,10         ;Adr. des ANZW (im DB 222
                          DW 10 + DW11)
```

```
DB 222
DW 10              ;
DW 11              ; ANZW
```

Indirekte Parametrierung von SSNR und BLGR (SYNCHRON)

```
A      DB X              DB aufschlagen
SPA    FB249
NAME   SYNCHR
SSNR   :KY 255, 100      ;Zeiger auf Parameterliste
BLGR   :KY 0,0           ;Irrelevant
   .
   :
DB     X

DW 100 KY 0,16           ;Parameter SSNR
DW 101 KY 0,6            ;BLGR
```

# Indirekte Parametrierung von Quell- und Zielangaben

**Parameter**

Parameter bei indirekter Parametrierung XX und RW

a) bei "XX"

DB-NR wenn High-Byte = 0

| | | |
|---|---|---|
| QANF + | 0 KC | QTYP/Ztyp jedoch nicht XX, RW |
| | 1 KY | DBNR DB-NR. bei "Typ" DB |
| | 2 KF | QANF/ZANF Anfangsadresse |
| | 3 KF | QLAE/ZLAE Länge |

b) bei "RW" (READ/WRITE)

DB-NR = DBNR wenn High-Byte = 0

| | | | |
|---|---|---|---|
| QANF + | 0 KC | QTYP jedoch nicht XX, RW | Beschreibung "Datenquelle" |
| | 1 KY | DBNR DB-NR. bei "Typ" DB | |
| | 2 KF | QANF Quell-Anfangsadresse | |
| | 3 KF | QLAE Quell-Länge | |
| | 4 KL | ZTYP jedoch nicht XX und WR | Beschreibung "Datenziel" |
| | 5 KY | DBNR DB-NR bei "ZTYP" DB | |
| | 6 KF | ZANF Ziel-Anfangsadresse | |
| | 7 KF | ZLAE Ziel-Länge | |

Bei Quell-/Ziel-Typ "AS" (Absolutadresse sind im Datenformat KH auch Adressen bis FFFFh (65535) parametrierbar.

# Tabelle über die möglichen QTYP/ZTYP - Parameter

| QTYP/ZTYP<br>Beschreibung | NN | XX | RW | DB | MB | AB |
|---|---|---|---|---|---|---|
| | keine Quell-/Zielparameter am Baustein; Parameter müssen auf dem CP vorhanden sein. | Indirekte Adressierung Parameter sind im (mit DBNR und QANF spez.) DB hinterlegt. | Indirekte Adressierung ohne Datenaustausch; Quell-/ Zielparameter sind in einem DB hinterlegt. | Quell-/Zieldaten aus/im DB im Hauptspeicher. | Quell-/Zieldaten aus/im Merkerbereich. | Quell-/Zieldaten aus/im Prozessabbild der Ausgänge (PAA). |
| DBNR<br>Bedeutung<br><br><br><br><br>erlaubter Bereich | irrelevant | DB, in dem die Quell-/Zielparameter hinterlegt sind.<br><br><br><br>2...255 | DB in dem die Quell-/Zielparameter hinterlegt sind.<br><br><br><br>2...255 | DB, aus dem die Quelldaten entnommen werden oder in den die Zieldaten transferiert werden.<br><br>2...255 | irrelevant | irrelevant |
| QANF/<br>ZANF<br>Bedeutung<br><br><br><br><br>erlaubter Bereich | irrelevant | DW-Nr., ab der die Parameter hinterlegt sind.<br><br><br><br>0...2047 | DW-Nr., ab der die Parameter hinterlegt sind.<br><br><br><br>0....2047 | DW-Nr., ab der die Daten entnommen oder eingeschrieben werden.<br><br><br><br>0...2047 | Merkerbyte-Nr. ab der die Daten entnommen oder eingeschrieben werden.<br><br><br><br>0...255 | Ausgangs-Byte-Nr., ab der die Daten entnommen oder eingeschrieben werden.<br><br><br><br>0...127 |
| QLAE/<br>ZLAE<br>Bedeutung<br><br>erlaubter Bereich | irrelevant | irrelevant | irrelevant | Länge des Quell-/Zieldatenblocks in Worten.<br><br>1...2048 | Länge des Quell-/Zieldatenblocks in Bytes.<br><br>1...255 | Länge des Quell-/Zieldatenblocks in Bytes.<br><br>1...128 |

*Fortsetzung ...*

*... Fortsetzung*

| QTYP/ ZTYP Beschreibung | EB | PB | ZB | TB | AS |
|---|---|---|---|---|---|
| | Quell- Zieldaten aus /in Prozess-abbild der Eingänge (PAE). | Quell- Zieldaten aus /in Peripherie-baugruppen. Bei Quell-daten Ein-gabebaugr., bei Zieldaten Ausgabe-baugr.. | Quell- /Zieldaten aus /in Zähler-zellen. | Quell- /Zieldaten aus /in Zeiten-zellen. | Quell- /Zieldaten aus /in absolut adressierten Speicher-zellen. |
| DBNR Bedeutung | irrelevant | irrelevant | irrelevant | irrelevant | irrelevant |
| QANF/ ZANF Bedeutung | Eingangsbyte-Nr. ab der die Daten ent-nommen oder eingeschrie-ben werden. | Peripheriebyte -Nr. ab der die Daten ent-nommen oder eingeschrie-ben werden. | Nummer der Zählerzelle, ab der die Daten ent-nommen oder eingeschrie-ben werden. | Nummer der Zeitenzelle, ab der die Daten entnommen oder einge-schrieben werden. | absolute An-fangsadresse, ab der die Daten ent-nommen oder eingeschrie-ben werden. |
| erlaubter Bereich | 0...127 | 0...127<br>digit. Peripher.<br>127...255<br>anal. Peripher. | 0...127 | 0...127 | 0...+32767<br>-32768 |
| QLAE/ ZLAE Bedeutung | Länge des Quell-/Ziel-datenblocks in Bytes. | Länge des Quell-/Ziel-datenblocks in Bytes. | Länge des Quell-/Ziel-datenblocks in Worten (Zählerzelle = 1 Wort). | Länge des Quell-/Ziel-datenblocks in Worten (Zeitenzelle = 1 Wort). | Länge des Quell-/Ziel-datenblocks in Worten. |
| erlaubter Bereich | 1...128 | 1...256 | 1...128 | 1..128 | 1...32767 |

# Aufbau des Anzeigenwortes

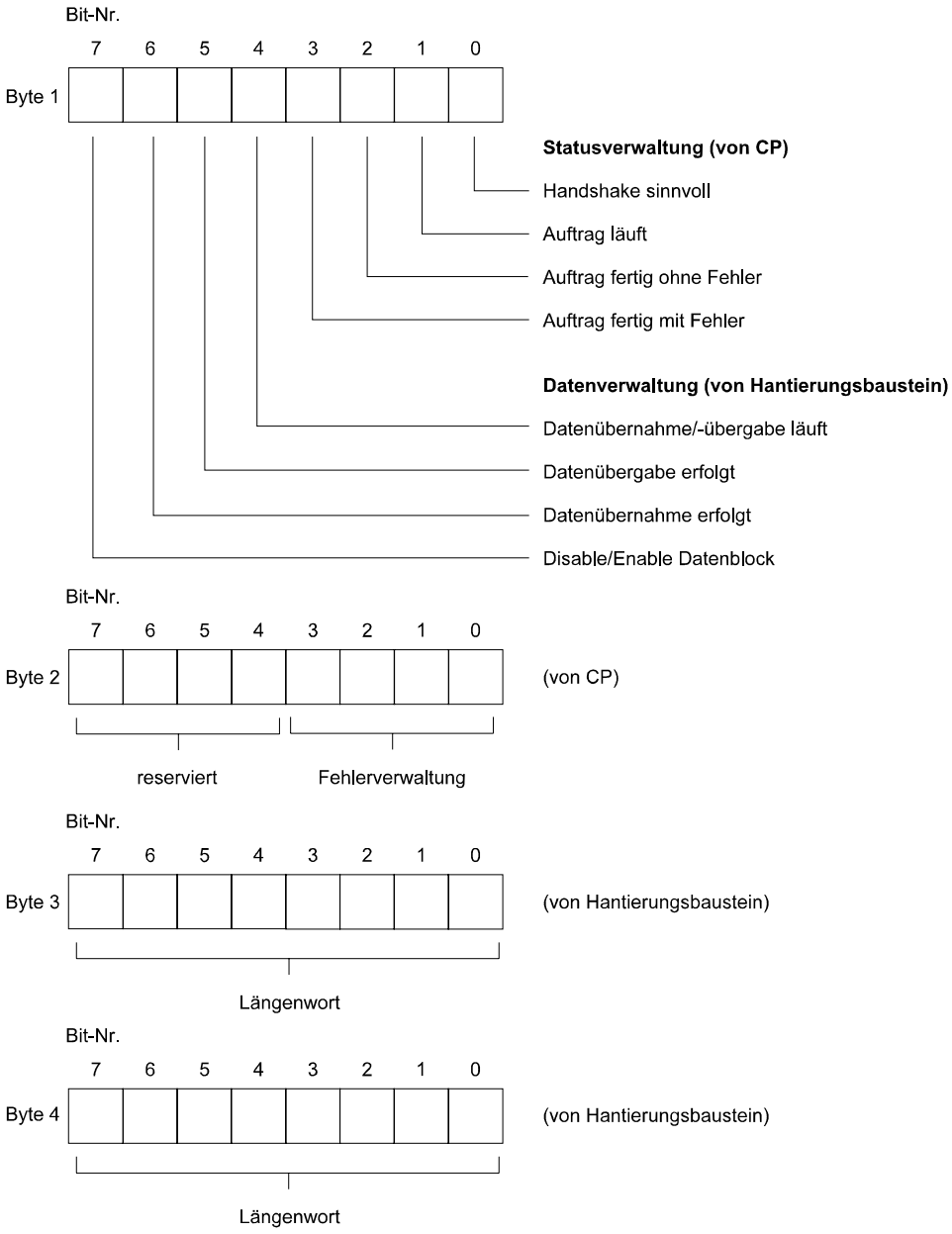
**Status- und Fehleranzeigen**

Status und Fehleranzeigen liefern die Hantierungsbausteine:

- über das Anzeigenwort ANZW (Informationen zur Auftragsbearbeitung),
- über das Parametrierfehlerbyte PAFE (Anzeige einer fehlerhaften Auftragsparametrierung).

**Inhalt und Aufbau Anzeigenwort ANZW**

Das Anzeigenwort hat den folgenden prinzipiellen Aufbau:



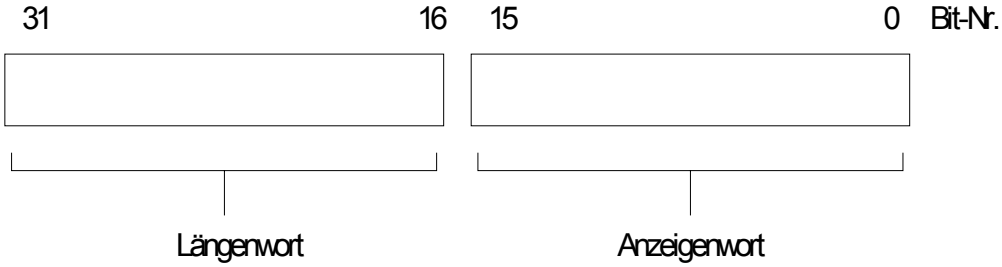
# Status und Fehleranzeige im Anzeigenwort

**Funktions-beschreibung**

Das "Anzeigenwort" zeigt den Zustand für einen bestimmten Auftrag auf einem CP an.

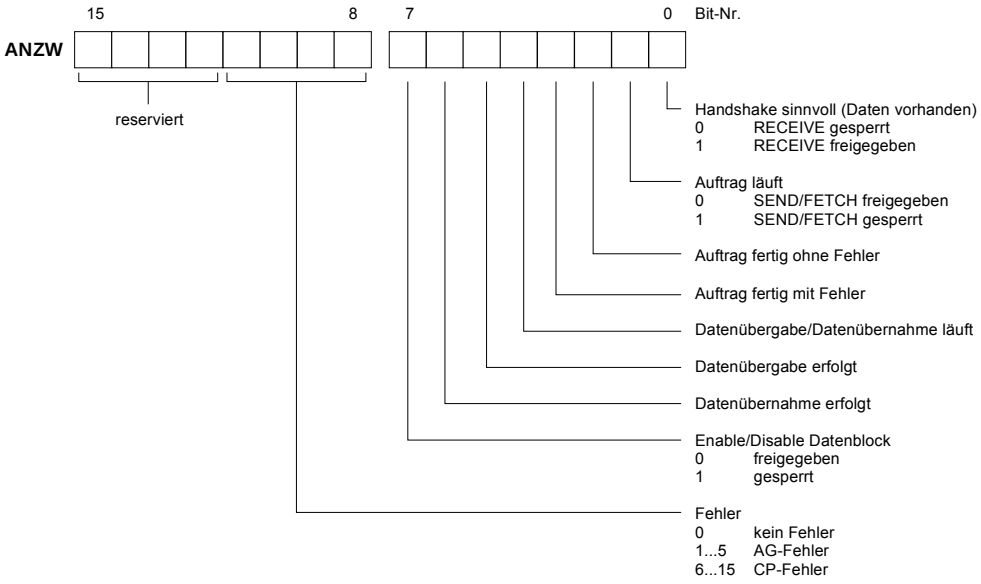
Im SPS-Programm sollte für jeden Auftrag ein eigenes "Anzeigenwort" für jeden definierten Auftrag bereitgestellt werden.

Das "Anzeigenwort" hat folgenden Aufbau:




Im Längenwort hinterlegen die Hantierungsbausteine (SEND, RECEIVE) die für den entsprechenden Auftrag bereits transferierten Daten; empfangene Daten in Empfangsaufträgen; bereits gesendete Daten in Sendeaufträgen.

Die Anzeige im "Längenwort" erfolgt immer in Bytes und absolut.



# Anzeigenwort

**Bit 0** **Handshake sinnvoll**

Setzen: Durch die Anschaltung entsprechend der "Löschen"-Anzeige im Auftragsstatus-Bit. Handshake sinnvoll (=1) wird beim RECEIVE-Baustein genutzt. (Telegramm vorhanden bei PRIO 1 oder RECEIVE-Anstoß möglich bei PRIO 2/3)

Auswerten: Durch den RECEIVE-Baustein: Nur wenn das Bit gesetzt, leitet der RECEIVE den Handshake mit dem CP ein. Durch die Anwendung: Für RECEIVE-Anfrage (Abfrage, ob Telegramm vorhanden bei PRIO 1).

**Bit 1** **Auftrag läuft**

Setzen: Durch die Anschaltung, wenn Auftrag an CP erteilt.

Löschen: Durch die Anschaltung, wenn ein Auftrag abgearbeitet ist (z.B. Quittung eingetroffen).

Auswerten: Durch die Hantierungsbausteine: Ein neuer Auftrag wird nur erteilt, wenn der "alte" Auftrag abgearbeitet ist.

Durch den Anwender: um zu erfahren, ob das Triggern eines neuen Auftrags sinnvoll ist.

**Bit 2** **Auftrag fertig ohne Fehler**

Setzen: Durch die Anschaltung, wenn der entsprechende Auftrag ohne Fehler abgeschlossen wurde.

Löschen: Durch die Anschaltung, wenn der Auftrag erneut ausgelöst wird.

Auswerten: Durch den Anwender zur Prüfung, ob der Auftrag fehlerlos abgeschlossen wurde.

**Bit 3** **Auftrag fertig mit Fehler**

Setzen: Durch die Anschaltung, wenn der entsprechende Auftrag mit Fehler abgeschlossen wurde. Die Fehlerursache ist dann im High-Teil des Anzeigenwortes verschlüsselt.

Löschen: Durch die Anschaltung, wenn der Auftrag erneut ausgelöst wird.

Auswerten: Durch den Anwender: Zur Prüfung, ob der Auftrag mit Fehler abgeschlossen, wurde. Ist die Kennung "Auftrag fertig mit Fehler" gesetzt, steht im High-Byte des Anzeigenwortes die Fehlerursache.

- **Datenverwaltung Byte 1, Bit 4 bis Bit 7**

Hier ist verschlüsselt, ob der Datentransfer für den Auftrag noch läuft oder ob die Datenübergabe bzw. Datenübernahme bereits abgeschlossen ist. Mit dem Bit "Enable / Disable" kann der Datentransfer für den Auftrag gesperrt werden. (Disable = 1; Enable = 0).

**Bit 4** **Datenübernahme / Datenübergabe läuft**

Setzen: Durch die Hantierungsbausteine SEND, RECEIVE, wenn die Übergabe/Übernahme für einen Auftrag begonnen wurde, z.B. wenn Daten über die ALL-Funktion (DMA-Ersatz) ausgetauscht werden, der Anstoß jedoch mit SEND-DIREKT erfolgte.

Löschen: Durch die Hantierungsbausteine SEND, RECEIVE, wenn der Datenaustausch für einen Auftrag beendet ist (letzter Teilblock übertragen).

Auswerten: Durch den Anwender: Während der Datenübertragung CP<< >>AG darf der Anwender den Datensatz eines Auftrags nicht mehr verändern. Bei PRIO 0/1 Aufträgen ist dies unkritisch, da hierbei der Datenaustausch in einem Baustein-Durchlauf erledigt werden kann. Größere Datenmengen können jedoch nur in Blöcken übertragen werden, wobei diese Blockung über mehrere AG-Zyklen verteilt wird. Zur Wahrung der Datenkonsistenz ist zu prüfen ob der Datenblock gerade übertragen wird, bevor dessen Inhalt geändert wird.

**BIT 5** **Datenübergabe erfolgt**

Setzen: Durch den Hantierungsbaustein SEND, wenn die Datenübergabe für einen Auftrag erfolgt ist.

Löschen: Durch den Hantierungsbaustein SEND, wenn für einen neuen Auftrag (neue Triggerung) mit dem Transfer der Daten begonnen wurde.

Durch den Anwender: Wenn die Auswertung erfolgte (Flankenbildung).

Auswerten: Durch den Anwender: Mit diesem Bit ist zu ermitteln, ob der Datensatz für einen Auftrag schon auf den CP übertragen wurde bzw. wann ein neuer Datensatz für einen laufenden Auftrag (z.B. zyklische Übertragung) bereitgestellt werden kann.

**Bit 6** **Datenübernahme erfolgt**

Setzen: Durch den Hantierungsbaustein RECEIVE, wenn die Übernahme von Daten für einen Auftrag abgeschlossen wurde.

Löschen: Durch den Hantierungsbaustein RECEIVE, wenn für einen neuen Auftrag (neue Triggerung) mit dem Transfer der Daten ins AG begonnen wurde. Durch den Anwender, wenn die Auswertung erfolgt (Flankenbildung).

Auswerten: Durch den Anwender: Mit diesem Bit kann der Anwender ermitteln, ob der Datensatz eines Auftrags schon auf das AG übertragen wurde bzw. wann ein neuer Datensatz für einen laufenden Auftrag ins AG transferiert wurde.

**Bit 7** **Disable / Enable Datenblock**

Setzen: Durch den Anwender, um das Beschreiben eines Bereichs durch den RECEIVE-Baustein bzw. das Auslesen aus einem Bereich durch den SEND-Baustein zu verhindern (nur beim 1. Datenblock).

Löschen: Durch den Anwender, um den zugehörigen Datenbereich freizugeben.

Auswerten: Durch die Hantierungsbausteine SEND und RECEIVE. Ist das Bit 7 gesetzt, führen die Bausteine keinen Datenverkehr durch, sondern melden dem CP den Fehler.

- **Fehlerverwaltung Byte 2, Bit 0 bis Bit 3**

Hier werden die Fehleranzeigen des Auftrags angezeigt. Diese Fehleranzeigen sind nur gültig, wenn auch gleichzeitig das Bit "Auftrag fertig mit Fehler" im Statusbit gesetzt ist.

Folgende Fehlermeldungen können ausgegeben werden:

0 **kein Fehler**

Sollte das Bit "Auftrag fertig mit Fehler" gesetzt sein, so hat der CP143 H1 / TCP/IP die Verbindung neu aufbauen müssen wie z.B. nach einem Neustart oder RESET.

1 **falscher Q/ZTYP am HTB**

Auftrag wurde mit falscher TYP-Kennung parametriert

2 **Bereich im AG nicht vorhanden**

Beim Anstoß des Auftrags wurde ein falscher DB (DBNR) parametriert.

3 **Bereich im AG zu klein**

Die Summe aus Q/ZANF und Q/ZLAE überschreitet die Bereichsgrenzen. Die Bereichsgrenze wird bei Datenbausteinen durch die Bausteingröße bestimmt. Bei Merkern, Zeiten, Zählern usw. ist die Bereichsgröße AG-abhängig.

4 **QVZ-Fehler im AG**

Mit dem Quell- bzw. Zielparameter wurde ein Bereich im AG angegeben, dessen Speicher defekt oder nicht bestückt ist. Der QVZ-Fehler kann nur bei Q/ZTYP AS, PB, QB oder bei Speicherdefekten auftreten.

5 **Fehler beim Anzeigenwort**

Das parametrierte Anzeigenwort kann nicht bearbeitet werden. Dieser Fehler tritt auf, wenn mit ANZW ein Datenwort bzw. Doppelwort angegeben wurde, das sich nicht oder nicht mehr in dem spezifizierten Datenbaustein befindet d.h. DB zu klein oder nicht vorhanden.

6 **kein gültiges ORG-Format**

Das Datenziel bzw. die Datenquelle ist weder beim Hantierungsbaustein (Q/TYP="NN") noch im Verbindungsbaustein angegeben.

7 **Reserviert**

8 **keine freien Transportverbindungen**

Die Transportverbindungskapazitäten sind überschritten. Löschen Sie unnötige Verbindungen.

9 **Remote-Fehler**

Bei einem READ/WRITE-Auftrag ist ein Fehler im Kommunikationspartner aufgetreten.

A **Verbindungsfehler**

Die Verbindung für einen Auftrag ist nicht bzw. noch nicht aufgebaut. Der Fehler verschwindet, sobald eine Verbindung aufgebaut werden kann. Sind alle Verbindungen des CPs unterbrochen, so deutet dies auf eine Defekt der Baugruppe oder des Buskabels hin. Der Fehler kann auch durch eine fehlerhafte Parametrierung ausgelöst werden, wie z.B. fehlerhafte Adressierung.

B **Handshakefehler**

Dies kann ein Systemfehler sein oder die Datenblockgröße ist zu groß gewählt.

C **Anstoßfehler**

Zum Anstoß des Auftrags wurde ein falscher Hantierungsbaustein benutzt, oder ein zu großer Datenblock übergeben.

D **Abbruch nach RESET**

Hier handelt es sich um eine Betriebsmeldung. Bei Priorität 1 und 2 ist die Verbindung unterbrochen und wird neu aufgebaut, sobald sich der Kommunikationspartner auf eine neue Verbindung eingestellt hat. Bei Priorität 3 Verbindungen ist die Verbindung gelöscht, ein neuer Anstoß ist möglich.

E **Auftrag mit Urladefunktion**

Dies ist eine Betriebsmeldung. Der Auftrag ist ein READ/WRITE-PASSIV und kann vom AG aus nicht gestartet werden.

F **Auftrag nicht vorhanden**

Der angesprochene Auftrag ist nicht auf dem CP143 H1 / TCP/IP parametriert. Dieser Fehler kann auftreten wenn SSNR/A-NR Kombination im Hantierungsbaustein falsch oder kein Verbindungsbaustein eingetragen ist.

Die Bits 4 bis 7 von Byte 2 sind für Erweiterungen reserviert.

**Status- und Fehleranzeigen**

### Wichtige Status- und Fehleranzeigen der CPU 24x NET

Im Folgenden sind wichtige Status- und Fehlermeldungen aufgeführt, die im "Anzeigenwort" erscheinen können. Die Darstellung hierbei erfolgt in "HEX"-Mustern, wie man sie auch mit der Status / Steuern-Var-Testfunktion mit dem PG im AG beobachten kann. Das Zeichen X steht für "nicht bestimmt" bzw. für "irrelevant"; Nr. ist die Fehlernummer.

### Mögliche Anzeigenworte

*Anzeigenwort: X F X A*

Die Fehlerkennung "F" besagt, dass der entsprechende Auftrag auf dem CP 143 nicht definiert ist. Die Statuskennung A bewirkt, dass der Auftrag gesperrt ist ( für SEND / FETCH und RECEIVE).

*Anzeigenwort: X A X A*

Die Fehlerkennung "A" zeigt an, dass die Verbindung des Kommunikationsauftrags nicht bzw. noch nicht aufgebaut ist. Mit der Statuskennung "A" ist sowohl der SEND als auch der RECEIVE und FETCH gesperrt.

*Anzeigenwort: X 0 X 8*

Die Verbindung ist neu aufgebaut (z.B. nach einem CP-Neuanlauf), der SEND ist freigegeben (SEND-Kommunikationsauftrag).

*Anzeigenwort: X 0 X 9*

Die Verbindung ist neu aufgebaut, der RECEIVE ist freigegeben. (RECEIVE-Kommunikationsauftrag).

*Anzeigenwort: X 0 2 4*

Der SEND ist ohne Fehler abgearbeitet worden, die Daten wurden übertragen.

*Anzeigenwort: X 0 4 5*

Der RECEIVE ist ohne Fehler abgearbeitet worden, die Daten sind auf dem AG angekommen.

Anzeigenwort: X 0 X 2

Der SEND-, RECEIVE-, READ- bzw. WRITE-Auftrag läuft. Bei SEND hat sich der Partner noch nicht auf den RECEIVE eingestellt. Bei RECEIVE hat der Partner noch kein SEND abgesetzt.

In den folgenden Tabelle sind die wichtigsten Anzeigenwortzustände aufgeführt:

Anzeigen bei SEND

| Zustand unter H1 | Prio 0/1 | Prio 2 | Prio 3/4 |
|---|---|---|---|
| Zustand unter TCP/IP | Prio 1 | Prio 2 | Prio 3 |
| nach Neustart | 0 A 0 A | 0 A 0 A | 0 0 0 8 |
| nach Verbindungsaufbau | X 0 X 8 | X 0 X 8 | ..... |
| nach Anstoß | X 0 X 2 | X 0 X 2 | X 0 X 2 |
| fertig ohne Fehler | X 0 2 4 | X 0 2 4 | X 0 2 4 |
| fertig mit Fehler | X Nr X 8 | X Nr X 8 | X Nr X 8 |
| nach RESET | X D X A | X D X A | X D X 8 |

Anzeigen bei RECEIVE

| Zustand unter H1 | Prio 0/1 | Prio 2 | Prio 3/4 |
|---|---|---|---|
| Zustand unter TCP/IP | Prio 1 | Prio 2 | Prio 3 |
| nach Neustart | 0 A 0 A | 0 A 0 A | 0 0 0 1 |
| nach Verbindungsaufbau | X 0 X 4 | X 0 0 9 | ..... |
| nach Anstoß | X 0 X 2 | X 0 X 2 | X 0 X 2 |
| Telegramm da | X 0 X 1 | ..... | ..... |
| fertig ohne Fehler | X 0 4 1 | X 0 4 5 | X 0 4 5 |
| fertig mit Fehler | X Nr X8 | X Nr X 9 | X Nr X 9 |
| nach RESET | X D X A | X D X A | X D X 9 |

Anzeigen bei READ/WRITE-AKTIV

| Zustand unter H1 | Prio 0/1 | Prio 2 | Prio 3/4 |
|---|---|---|---|
| Zustand unter TCP/IP | Prio 1 | Prio 2 | Prio 3 |
| nach Neustart | | 0 A 0 A | |
| nach Verbindungsaufbau | | X 0 0 8 | |
| nach Anstoß | | X 0 X 2 | |
| READ fertig | | X 0 4 4 | |
| WRITE fertig | | X 0 2 4 | |
| fertig mit Fehler | | X Nr X 8 | |
| nach RESET | | X D X A | |

Anzeigen bei SEND bzw. RECEIVE bei Hantierungsbaustein-Kennung "NN" (keine Quell- / Zielparameter)

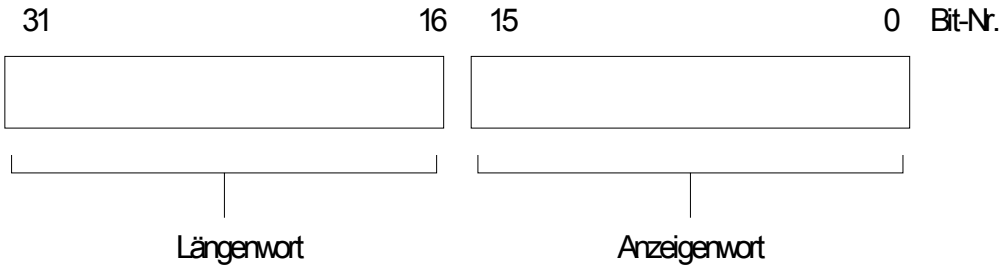
bei SEND

| Zustand unter H1 | Prio 0/1 | Prio 2 | Prio 3/4 |
|---|---|---|---|
| Zustand unter TCP/IP | Prio 1 | Prio 2 | Prio 3 |
| fertig ohne Fehler | X 0 0 4 | X 0 0 4 | X 0 0 4 |

bei RECEIVE

| Zustand unter H1 | Prio 0/1 | Prio 2 | Prio 3/4 |
|---|---|---|---|
| Zustand unter TCP/IP | Prio 1 | Prio 2 | Prio 3 |
| fertig ohne Fehler | X 0 0 4 | X 0 0 5 | X 0 0 5 |

# Längen - Wort




Im Längenwort hinterlegen die Hantierungsbausteine (SEND, RECEIVE) die für den entsprechenden Auftrag bereits transferierten Daten; empfangene Daten in Empfangsaufträgen; bereits gesendete Daten in Sendeaufträgen.

Die Anzeige im "Längenwort" erfolgt immer in Bytes und absolut.

- **Längenwort Byte 3 und Byte 4**

Im Längenwort hinterlegen die Hantierungsbausteine (SEND, RECEIVE) die Menge für den entsprechenden Auftrag bereits transferierten Daten, d.h. bei Empfangsaufträgen die bereits empfangene Datenmenge, bei Sendeaufträgen die bereits gesendete Datenmenge.

Beschreiben: Durch SEND, RECEIVE während des Datenaustausches. Das "Längen-Wort" wird errechnet aus:

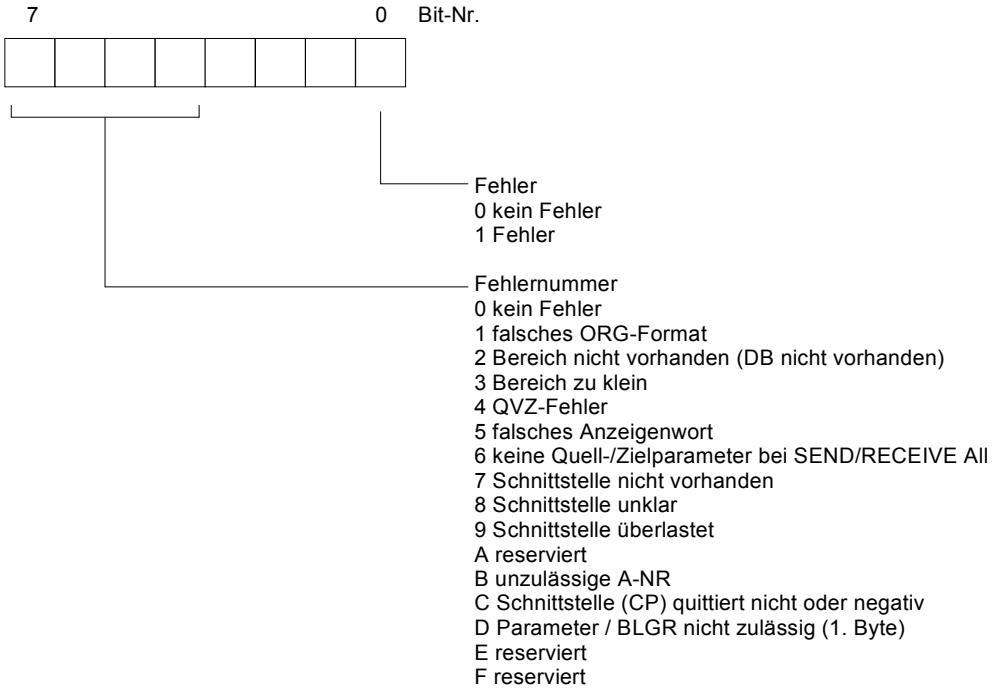
**aktuelle Übertragungsanzahl + Anzahl bereits ausgetauschter Daten**

Löschen: Durch Überschreiben bzw. mit jedem neuen SEND, RECEIVE, FETCH.

Wenn das Bit "Auftrag fertig ohne Fehler" bzw. "Datenübergabe/-übernahme erfolgt" gesetzt ist, steht im "Längen-Wort" die aktuelle Quell- bzw. Ziellänge.

Wenn das Bit "Auftrag fertig mit Fehler" gesetzt ist, beinhaltet das Längenwort die bis zum Fehlerfall übertragene Datenanzahl.

# Aufbau des Parametrierfehler - Anzeigenbytes



# Einstellbare Blockgrößen

Als Blockgröße kann eingestellt werden:

0 Der Baustein benutzt die Default-Parameter
(Bei der CPU 24x auf 64Byte eingestellt)

1 Blockung mit 16Byte

2 Blockung mit 32Byte

3 Blockung mit 64Byte

4 Blockung mit 128Byte

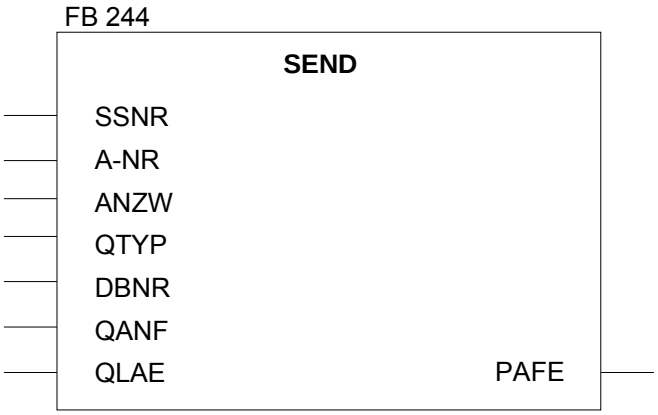
5 Blockung mit 256Byte

6 Blockung mit 512Byte

>6 und < 255 wie 0.

# FB 244 - SEND

Der SEND-Baustein dient zum Auslösen eines Auftrags zu einem CP mit oder ohne Datenübergabe.




**Parameter**

| | |
|---|---|
| SSNR: | Schnittstellennummer, Nummer der logischen Schnittstelle |
| A-NR: | auszulösender Auftrag der Schnittstelle d.h. Senden eines Telegramms starten. |
| ANZW: | Adresse des Anzeigenworts (Doppelwort), in dem die Abarbeitung des ausgelösten Auftrages angezeigt wird. |
| QTYP: | Art der Datenquelle aus der die Daten übergeben werden. |
| DBNR: | Nummer des Datenbausteins bei QTYP XX, RW (Read/Write), DB |
| QANF: | "Relative Anfangsadresse" der Datenquelle. |
| QLAE: | Anzahl der Quelldaten (in Bytes oder Worten). |
| PAFE: | Fehleranzeige bei Parametrierungsfehler. |

**SEND-ALL**

*Beschreibung der SEND-ALL-Funktion*

Für die SEND-ALL Funktion (Auftragsnummer 0) benötigt der Baustein nur die Parameter: SSNR, A-NR = 0, ANZW und PAFE.

Alle anderen Parameter sind bei diesem Auftrag irrelevant. Die Adresse des Anzeigenworts, die Angabe des Datentyps, der Anzahl und der Anfangsadresse der Daten muss bei der SEND-ALL - Funktion vom CP über den Kommunikationsbereich zur Verfügung gestellt werden.

Im Anzeigenwort, das dem betreffenden Auftrag zugeordnet ist, werden die Bits "ENABLE/DISABLE", "Datenübergabe erfolgt" sowie "Datenübergabe läuft" ausgewertet oder beeinflusst. Die Anzahl der übertragenen Daten zu einem Auftrag zeigt SEND-ALL in dem ANZW folgenden Daten- bzw. Merkerwort an.

*Im Baustein-Anzeigenwort* (Anzeigenwort, das im SEND-ALL-Baustein parametriert ist) wird die aktuelle Auftragsnummer hinterlegt (0 bedeutet Leerdurchlauf). Die Anzahl der übertragenen Daten zu einem Auftrag zeigt der SEND-ALL in dem Datenwort an, das dem Anzeigenwort folgt.

Der Aufruf der SEND-ALL-Funktion (d.h. der Aufruf des Send-Bausteins mit der ALL-Parametrierung) muss mindestens einmal im AG-Zyklus pro Schnittstelle erfolgen, wenn:

- der CP selbständig Daten von einem AG anfordern kann.
- Ein CP-Auftrag mit einem SEND-DIREKT angestossen wird, der CP die Daten zu diesem Auftrag jedoch erst über die "Hintergrundkommunikation" beim AG anfordert.
- Die Anzahl der Daten, die mit einem SEND-DIREKT dem CP übergeben werden sollen, größer der eingestellten Blockgröße ist.

**SEND-DIRECT**

Beschreibung der SEND-DIREKT-Funktion

Für die "DIREKT"-Funktion benötigt der Baustein die Parameter:

- Schnittstellennummer
- Auftragsnummer >< 0
- Angabe des Anzeigenwortes
- Angabe des Fehlerbytes "PAFE"
- Quelltyp mit DBNR
- Quell-Anfangsadresse
- Anzahl der Quelldaten.

Der "SEND-DIREKT"-Baustein wird im Normalfall im zyklischen Teil des Anwenderprogramms aufgerufen. Die Einbindung des Bausteins im Interrupt oder Weck-Programmteil ist zwar möglich, das Anzeigenwort (ANZW) kann hierbei jedoch nicht zyklisch aktualisiert werden (muss durch den CONTROL-Baustein übernommen werden).

Der Verbindungsaufbau mit dem CP wird für die Datenübergabe und für die Aktivierung eines Send-Anstoßes nur dann aufgenommen wenn:

- dem FB VKE "1" übergeben wurde.
- der CP den Auftrag freigegeben hat. (Bit "Auftrag läuft" im ANZW =0).

Im Leerlauf des Bausteins (bei Übergabe von VKE "0") wird nur das Anzeigenwort aktualisiert.

Ist im QTYP-Parameter die Kennung "NN" eingetragen, werden die Quellparameter von CP benutzt. Fehlen auch diese Parameter, wird der Auftrag mit einer Fehlermeldung beendet.

Kann der CP die Daten direkt übernehmen, überträgt der SEND-Baustein die angeforderten Daten in einem Zug zum CP. Signalisiert der CP jedoch, dass er nur die Parameter des Auftrages wünscht oder die Anzahl der zu übergebenden Daten ist zu groß, werden dem CP nur die Parameter (QTYP/QLAE usw.), bzw. die Parameter mit dem ersten Datenblock übergeben. Die Daten oder der Folgeblock zu diesen Aufträgen fordert der CP über die SEND ALL - Funktion beim AG an. Hierzu ist es jedoch notwendig, dass mindestens einmal im AG-Zyklus der SEND ALL Baustein aufgerufen wird.

Die Bedienoberfläche ist in allen "Anstoßarten" für den Anwender der Bausteine gleich, nur der Zeitpunkt der Datenübergabe ist bei den zuletzt genannten Fällen um mindestens einen AG-Zyklus verschoben.

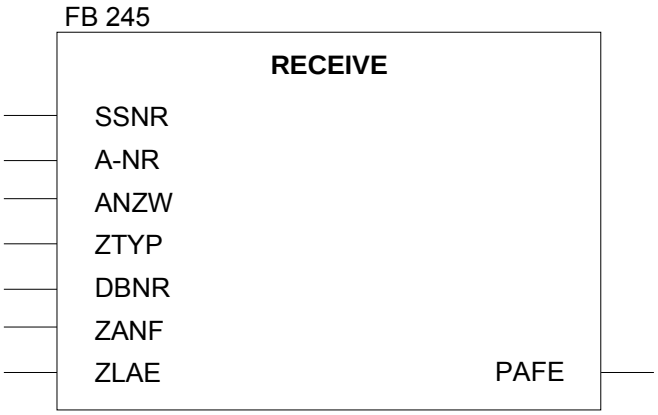
**WRITE-Funktion**

*Beschreibung der WRITE-Funktion*

Die indirekt angegebenen Quell- und Zielparameter werden bei der Parametrierung mit QTYP = "RW" von "SEND" zum CP übertragen. Hier werden die Zielparameter zusammen mit den Nutzdaten (die über SEND ALL angefordert wurden) zum Kommunikationspartner gesendet ("WRITE"-Funktion).

# FB 245 - RECEIVE

Der RECEIVE-Baustein dient dem Abfragen oder dem Starten eines RECEIVE-Auftrages auf einem CP mit oder ohne Datenübernahme.




**Parameter**

| | |
|---|---|
| SSNR: | Schnittstellennummer, Nummer der logischen Schnittstelle |
| A-NR: | RECEIVE-Auftrag der Schnittstelle Abfrage nach einem Empfangstelegramm mit Datenübergabe. |
| ANZW: | Adresse des Anzeigenworts (Doppelwort), in dem die Abarbeitung des Auftrages angezeigt wird. |
| ZTYP: | Art des Datenziels in das die Daten übergeben werden sollen. |
| DBNR: | Nummer des DBs bei ZTYP XX, DB |
| ZANF: | "Relative Anfangsadresse" des Datenziels |
| ZLAE: | Anzahl der Daten die übertragen werden sollen. |
| PAFE: | Fehleranzeige bei Parametrierungsfehler. |

**RECEIVE-ALL**

*Beschreibung der RECEIVE-ALL-Funktion*

Der Baustein benötigt für die RECEIVE-ALL-Funktion (Auftragsnummer "0") nur die Schnittstellennummer, die Auftragsnummer = 0, den Ausgang "PAFE" und das Anzeigenwort. Alle anderen Parameterangaben sind in dieser Betriebsweise irrelevant. Die Adresse des Anzeigenworts, die Angabe des Typs, der Anfangsadresse und der Anzahl der Zieldaten stellt bei der RECEIVE ALL-Funktion der CP über den Kommunikationsbereich dem FB zur Verfügung. Im Anzeigenwort, das dem zu bearbeitenden Auftrag zugeordnet ist, werden die Bits "Enable/Disable", "Datenübernahme erfolgt" sowie "Datenübernahme/-übergabe läuft" ausgewertet oder beeinflusst und im Folgewort die "Empfangslänge" angezeigt.

Im Baustein- Anzeigenwort (parametriertes Anzeigenwort von RECEIVE ALL) wird die aktuelle Auftragsnummer hinterlegt, für den der RECEIVE ALL aktiv war. Im Leerlauf des RECEIVE ALL ist das Baustein-Anzeigenwort gelöscht.

Der RECEIVE ALL (das heißt der RECEIVE-Baustein mit der ALL-Parametrierung) ist immer dann mindestens einmal im AG-Zyklus aufzurufen, wenn:

- der CP selbständig Daten an das AG abgeben will.
- Mit einem RECEIVE-DIREKT ein Datenblock empfangen werden soll, der größer als die eingestellte Blockgröße ist.

Der Funktionsbaustein "RECEIVE-ALL" kann vom Anwender aufgerufen werden im:

- zyklischen Programmteil (z.B. im OB 1),
- zeitgesteuerten Programmteil (z.B. Weck-Baustein),
- Interrupt-Programmteil (Prozessalarme).

**RECEIVE-DIRECT**

Für die RECEIVE-DIRECT Funktion benötigt der Baustein folgende Parameter:

- SSNR Schnittstellennummer
- A-NR Auftragsnummer ><0
- ANZW Angabe des Anzeigenwortes
- PAFE Angabe des Fehlerbytes "PAFE"
- ZTYP Zieltyp evtl. mit DBNR
- ZANF Ziel-/Anfangsadresse
- ZLAE Anzahl der zu übernehmenden Daten

Im Normalfall wird der RECEIVE-Baustein im zyklischen Teil des Anwenderprogramms aufgerufen. Die Einbindung des Bausteins im Interrupt oder Weck-Programmteil ist ebenso möglich, dabei wird jedoch das Anzeigenwort nicht zyklisch aktualisiert. Diese Funktion muss dann der CONTROL-Baustein übernehmen.

Der Quittungsverkehr mit dem CP (das heißt ein Auftragsanstoß) wird vom RECEIVE-Baustein in der "DIREKT"-Funktion nur aufgenommen wenn:

- dem FB VKE "1" übergeben wurde und
- der CP den Auftrag freigeben hat. (Bit "Handshake sinnvoll" = 1).

Im "Leerlauf" des Bausteins wird nur das Anzeigenwort aktualisiert.

Der RECEIVE-(DIRECT)-Baustein verhält sich unterschiedlich je nach Art der Versorgung und der CP-Reaktion:

- Ist im ZTYP-Parameter die Kennung "NN" eingetragen, erwartet der Baustein die Zielparameter (Typ, Anfang, Länge des Zieldatenblocks) vom CP. Wird vom CP ein Parametersatz geliefert, obwohl der RECEIVE-Baustein selbst mit den Zielparametern versorgt wurde, (ZTYP><NN), haben die Parameterangaben am Baustein Priorität gegenüber dem Parametersatz vom CP.
- Große Datenmengen können nur in Blöcken übernommen werden. Hierzu ist es notwendig, solche Folgeblöcke mit dem RECEIVE-ALL ins AG zu übertragen. Der Aufruf des RECEIVE-ALL mindestens einmal im AG-Zyklus (pro CP-Schnittstelle) ist daher immer dann notwendig, wenn mit einem CP größere Datenblöcke ausgetauscht werden sollen. Ebenso ist die zyklische Einbindung des RECEIVE-ALL notwendig, wenn der CP den RECEIVE-DIREKT nur zur Freigabe eines Empfangstelegramms benutzt und die Daten über die "Hintergrundkommunikation" dem AG übergibt.
- Die Parametrierung mit ZTYP = "RW" ist beim RECEIVE nicht erlaubt.

# FB 246 - FETCH

Der FETCH-Baustein dient dem Auslösen eines "Holauftrags" auf einer Gegenstation.

Mit dem FETCH-Auftrag werden Daten-Quelle und -Ziel definiert und die Datenquelle an die Gegenstation übertragen.

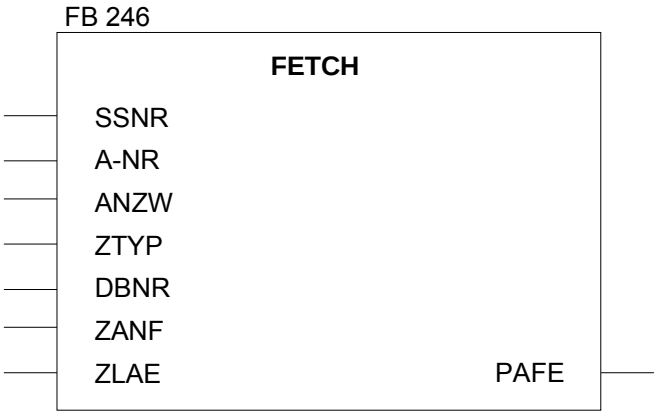
Bei der CPU 24x von VIPA erfolgt die Angabe von Quelle und Ziel indirekt über einen DB, dessen Nr Sie bei FETCH unter DBNR eintragen. Die Kennzeichnung der indirekten Angabe erfolgt über den ZTYP=RW.

Die Gegenstation stellt die Daten bereit und schickt diese über SEND-ALL an die anfordernde Station zurück. Über RECEIVE-ALL werden die Daten empfangen und in Ziel abgelegt.

Die Aktualisierung des Anzeigenworts erfolgt über FETCH bzw. CONTROL.

Der Quittungsverkehr für den Anstoß des FETCH wird nur aufgenommen, wenn

- dem Baustein VKE "1" übergeben
- im entsprechenden CP-Anzeigenwort die Funktion freigeben wurde (Auftrag läuft = 0).




**Parameter**

| | |
|---|---|
| SSNR: | Schnittstellennummer, Nummer der logischen Schnittstelle |
| A-NR: | auszulösender FETCH- oder READ-Auftrag |
| ANZW: | Anzeigenwort (Adresse des Anzeigenwortes). |
| ZTYP: | Art des Datenziels (DB, Merker, usw.) |
| DBNR: | Nummer des DBs bei ZTYP XX, RW, DB |
| ZANF: | "Relative Anfangsadresse" des Datenziels |
| ZLAE: | Anzahl der Daten die übernommen werden sollen. |
| PAFE: | Fehleranzeige bei Parametrierungsfehler. |

**Hinweis**

Nähere Angaben finden Sie unter "Indirekte Parametrierung".

# FB 247 - CONTROL

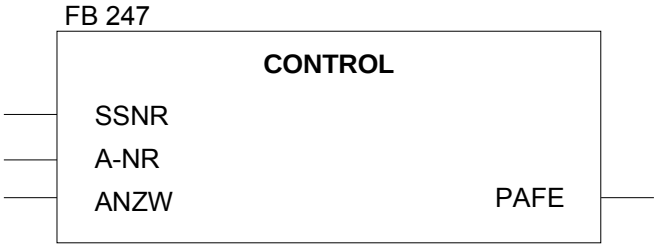
Der CONTROL-Baustein hat folgende Aufgaben:

- Aktualisierung des Anzeigenworts (Auftragsnummer >< 0 parametriert).
- Abfrage, ob ein bestimmter Auftrag des CP zur Zeit "tätig" ist, z.B. Nachfrage nach einem Empfangstelegramm (Auftragsnr. >< 0 parametriert).
- Abfrage des CP, welcher Auftrag z.Z. bearbeitet wird (Auftragsnr. = 0 parametriert).

Der CONTROL-Baustein nimmt keinen Quittungsverkehr mit dem CP auf, sondern überträgt nur die Anzeigen aus dem "Auftragsstatus" zum parametrierten Anzeigenwort. Der Baustein ist nicht VKE abhängig und sollte im zyklischen Teil des Programms aufgerufen werden.

Bei der CONTROL-DIRECT-Funktion (Parameter A-NR ><0) wird das Anzeigenwort in der gleichen Weise aufgebaut und bearbeitet wie bei allen anderen "Hantierungsbausteinen" (siehe Beschreibung des Bausteinparameters ANZW).

Wird der Parameter "Auftragsnummer " mit 0 versorgt, überträgt der CONTROL-Befehl den Inhalt der Auftragsstatuszelle 0 zum LOW-Teil des Anzeigenworts. In die Auftragsstatuszelle 0 schreibt der CP die Nummer des aktuellen Auftrags (das heißt des Auftrags, der gerade bearbeitet wird, wie z. B. die Auftragsnummer eines Telegramms).



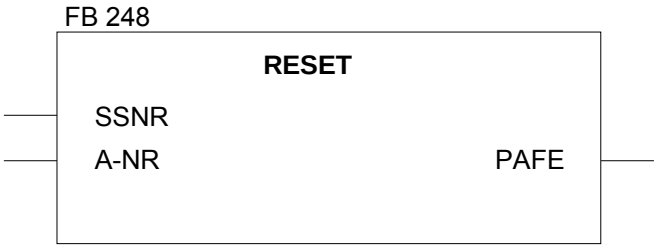

**Parameter**

| | | |
|---|---|---|
| SSNR: | Schnittstellennummer |
| A-NR: | Zu überwachender Auftrag des CPs z.B. Abfrage, ob der gestartete Auftrag mit oder ohne Fehler abgeschlossen wurde. |
| ANZW: | Anzeigenwort, in dem das Ergebnis der Abfrage dem Anwender mitgeteilt wird. |
| PAFE: | Fehlerbyte bei Parametrierungsfehlern. |

# FB 248 - RESET

Die RESET ALL-Funktion wird mit der Auftragsnummer 0 angewählt. Sie setzt alle Aufträge dieser logischen Schnittstelle zurück; z.B. löscht sie alle Auftragsdaten und bricht alle laufenden Aufträge ab. Mit einer "direkten" Funktion (Auftragsnummer >< 0) wird nur der angegebene Auftrag auf der logischen Schnittstelle rückgesetzt.

Der Baustein arbeitet VKE-abhängig und kann vom zyklischen, zeitgesteuerten oder alarmgesteuerten Programmteilen aus aufgerufen werden.




**Parameter**

| | |
|---|---|
| SSNR: | Schnittstellennummer |
| A-NR: | Nummer des rückzusetzenden Auftrages |
| PAFE: | Fehlerbyte bei Parametrierungsfehler |

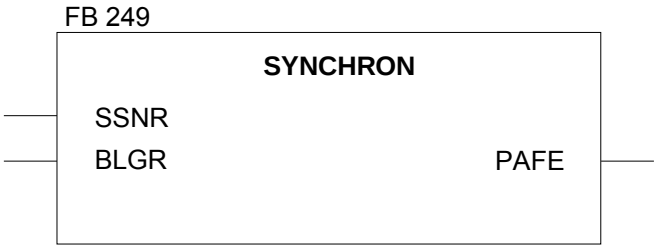
**Betriebsarten**

Der Baustein kennt folgende beiden Betriebsarten:

- RESET ALL
- RESET DIREKT.

# FB 249 - SYCHRON

Der Baustein stellt im AG-Anlauf die Synchronisation zwischen AG und CP her und ist daher in den Anlauf-OBs aufzurufen. Gleichzeitig wird der Übergabebereich der Schnittstelle gelöscht und voreingestellt, sowie die Blockgröße zwischen CP und AG ausgehandelt.




**Parameter**

| | |
|---|---|
| SSNR: | Schnittstellennummer |
| BLGR: | Blockgröße |
| PAFE: | Parametrierungsfehler |

**Blockgröße**

Als Blockgröße kann eingestellt werden:

| Blockgröße | |
|---|---|
| 0 | Standardwert (64Byte) |
| 1 | 16Byte |
| 2 | 32Byte |
| 3 | 64Byte |
| 4 | 128Byte |
| 5 | 256Byte |
| 6 | 512Byte |
| 7...254 | Standardwert wie 0 |
| 255 | 512Byte |

# Integrierte Sonder-Organisationsbausteine

Im Gegensatz zu den System-OBs, die zu programmieren sind, gelesen und verändert werden können, können die integrierten Sonder-OBs nur aufgerufen werden.

Nachfolgend sind alle integrierten OBs aufgeführt

**System-OBs**

| OB-Nr | Funktion |
|---|---|
| System-OBs | |
| OB 1 | zyklische Programmbearbeitung |
| OB 2 | Alarm: Alarmgenerierung durch die Digital- Eingabebaugruppe |
| OB 6 | Alarm ausgelöst durch internen Timer |
| OB 10 | zeitgesteuerte Programmbearbeitung (jeweils variabel: 10ms...10min) |
| OB 11 | zeitgesteuerte Programmbearbeitung (jeweils variabel: 10ms...10min) |
| OB 12 | zeitgesteuerte Programmbearbeitung (jeweils variabel: 10ms...10min) |
| OB 13 | zeitgesteuerte Programmbearbeitung (jeweils variabel: 10ms...10min) |
| OB 19 | bei Aufruf eines nicht geladenen Bausteins |
| OB 21 | bei manuellem Einschalten (STOP→RUN) |
| OB 22 | bei Spannungswiederkehr |
| OB 23 | Quittungsverzug bei Einzelzugriffen auf den Rückwandbus (z.B. L PB, LIR usw.) |
| OB 24 | Quittungsverzug beim Aktualisieren des Prozessabbilds |
| OB 27 | Substitutionsfehler |
| OB 32 | Transferfehler im DB oder bei EDB-Operation |
| OB 33 | Weckfehler |
| OB 34 | Batterieausfall |

Eine Beschreibung der System-OBs finden Sie im Kapitel "Einführung in die Programmiersprache".

**Sonder-OBs**

Folgende Sonder-OBs sind zur Zeit in die CPU 24x integriert und nachfolgend beschrieben:

| Art | Nr. | Funktion |
|---|---|---|
| OB | 31 | Zykluszeit neu starten |
| OB | 160 | Variable Zeitschleife |
| OB | 251 | PID-Regelalgorithmus |

# OB 31 - Zykluszeittriggerung

Durch einen "Zykluswächter" wird der zeitliche Ablauf einer Programmbearbeitung kontrolliert. Dauert eine Programmbearbeitung länger als die eingestellte Zyklusüberwachungszeit, z.B. 500ms, geht die CPU in STOP.

Dies kann z.B. eintreten bei:

- Überlänge des Steuerungsprogramms
- Programmierung einer Endlosschleife.

Durch Aufruf des OB 31 (SPA OB31) kann an einer beliebigen Stelle des Steuerungsprogramms der Zykluswächter nachgetriggert werden; d. h., die Zyklus-Überwachungszeit wird neu gestartet.

Die Zyklus-Überwachungszeit kann eingestellt werden:

- Im Systemdatenwort 96 (EAC0h)

oder

- im DB1.

# OB 160 - Variable Zeitschleife

Der OB 160 "simuliert" Operationslaufzeiten. Damit sind Sie unabhängig von den unterschiedlichen Operationslaufzeiten der verschiedenen CPUs und Sie können Wartezeiten einheitlich für alle CPUs programmieren.

**Vorgehensweise:** Vor dem Aufruf des OB 160 muss die Wartezeit in µs (Bereich: 10...65535 bzw. A0h...FFFFh) in den AKKU geladen werden.

**Beispiel:**

```
Es soll eine Wartezeit von einer Millisekunde programmiert
werden.

L   KF +1000
SPA OB 160
```

**Hinweis!**

Beachten Sie folgendes bei der Programmierung des OB 160:

Ein Prozessalarm (OB 2) und der Zeitalarm (OB 6) können die Wartezeit unterbrechen (vorausgesetzt, es wurde keine Alarmsperre (AS) programmiert).

Während der Unterbrechung läuft die Wartezeit nicht weiter! Ebenso wird die Laufzeit durch laufende PG/OP-Operationen verlängert.

Bei den eingestellten Zeiten handelt es sich also um Mindestzeiten!

Ein OB 10...13 kann den OB 160 **nicht** unterbrechen!

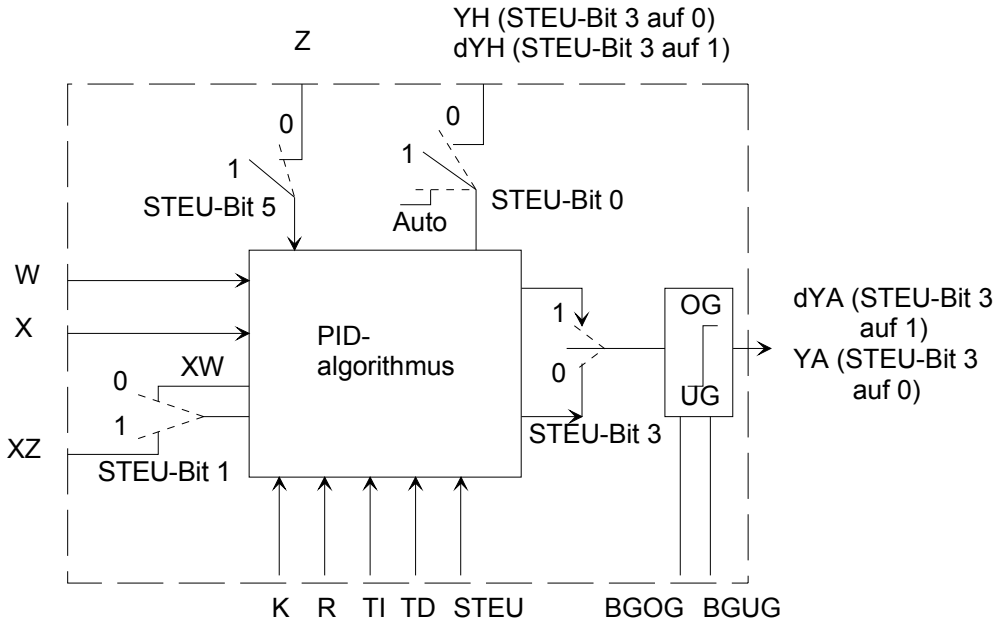
# OB 251 - PID-Regelalgorithmus

**Einführung**

Im Betriebssystem der Zentralbaugruppen ist ein PID Regelalgorithmus integriert, den der Anwender mit Hilfe des Organisationsbausteins OB 251 für seine Zwecke nutzen kann.

Vor dem Aufruf des OB 251 muss ein DB (Regler-DB) aufgeschlagen sein, der die Reglerparameter und sonstigen reglerspezifischen Daten enthält. Der PID-Algorithmus wird in einem bestimmten Zeitraster (Abtastzeit) aufgerufen und bildet die Stellgröße. Je genauer die Abtastzeit eingehalten wird, desto genauer kann der Regler seine Aufgaben erfüllen. Die im Regler-DB angegebenen Regelparameter müssen an die Abtastzeit angepasst sein. Im allgemeinen wird das Abtastzeitraster mit einem Zeit-OB (OB 10 bis OB 13) realisiert.

Zeit-OBs können im Aufrufintervall von 10 ms bis 10 min eingestellt werden. Die maximale Bearbeitungszeit des PID-Regelalgorithmus beträgt 2 ms.



**Legende**

| | | | | | |
|---|---|---|---|---|---|
| K | = | Proportionalbeiwert<br>K>0 positiver Regelsinn<br>K<= negativer Regelsinn | Y | = | Stellgröße |
| | | | dY | = | Stellinkrement |
| | | | YH | = | Stellwert bei man. Eing. |
| | | | dYH | = | Stellinkrement bei man. Eingabe |
| R | = | R-Parameter (i.a. 1000) | | | |
| TA | = | Abtastzeit | BGOG | = | Oberer Begrenzungsw. |
| TN | = | Nachstellzeit | BGUG | = | Unterer Begrenzungsw. |
| TV | = | Vorhaltezeit | X | = | Istwert |
| TI | = | TA/TN | Z | = | Störgröße |
| TD | = | TV/TA | XZ | = | Ersatzgröße für Regeldifferenz |
| STEU | = | Steuerwort | | | |
| W | = | Sollwert | YA | = | Reglerausgang: Stellgröße begrenzt |
| XW | = | Regeldifferenz | | | |
| | | | dYA | = | Reglerausgang: Stellinkrement begrenzt |

**Beschreibung**

Der quasikontinuierliche Regler ist für Regelstrecken ausgelegt, wie sie z.B. in der Verfahrenstechnik als Druck-, und Temperatur- oder Durchflussregelungen auftreten.

Mit der Größe "R" wird der Proportionalanteil des PID-Reglers eingestellt. Soll der Regler ein P-Verhalten zeigen, so wird bei den meisten Reglerentwurfsverfahren der Wert R = 1 verwendet.

Die einzelnen P-, I- und D-Anteile sind über ihre jeweiligen Parameter (R,TI und TD) abschaltbar, indem die betreffenden Datenwörter mit Null vorbesetzt werden. Damit können alle gewünschten Reglerstrukturen (z.B. PI, PD- oder PID-Regler) leicht realisiert werden.

Dem Differenzierer kann wahlweise die Regeldifferenz XW oder - über den XZ-Eingang - eine beliebige Störgröße oder der invertierte Istwert X zugeführt werden. Für einen invertierten Regelsinn muss ein negativer K-Wert vorgegeben werden.

Liegt die Stellinformation (dY oder Y) an einer Begrenzung, so wird der I-Anteil automatisch abgeschaltet, um eine Verschlechterung des Regelverhaltens zu vermeiden.

Die Schalterstellungen im Blockschaltbild werden bei der Parametrierung des PID-Reglers durch Setzen der zugehörigen Bits im Steuerwort "STEU" realisiert.

**Steuerwort STEU**

| Steuer bit | Name | Signal-zustand | Bedeutung |
|---|---|---|---|
| 0 | AUTO | 0 | **Handbetrieb**<br>Im Handbetrieb werden folgende Größen aktualisiert:<br>$X_{,k}$ $XW_{k-1}$und $PW_{k-1}$<br>$XZ_{,k}$, $XZ_{k-1}$ und $PZ_{k-1}$, wenn STEU-Bit =1<br>$Z_k$, und $Z_{k-1}$, wenn STEU-Bit 5 =0<br>Die Größe $dD_{k-1}$, wird = 0 gesetzt. Der Algorithmus wird nicht berechnet. |
| | | 1 | **Automatikbetrieb** |
| 1 | XZ EIN | 0 | Der Differenzierer wird $XW_{,k}$ zugeführt. Der XZ-Eingang bleibt unberücksichtigt. |
| | | 1 | Dem Differenzierer wird über den XZ-Eingang eine andere Größe, die nicht $XW_{,k}$ sein darf, zugeführt. |
| 2 | REG AUS | 0 | normale Reglerbearbeitung |
| | | 1 | Beim Aufruf des Reglers (OB 251) werden mit Ausnahme von K, R, TI, TD, BGOG, BGUG, $YH_{,k}$ und $W_{,k}$ alle anderen Größen (DW 18 bis DW 48) im Regler-DB einmal gelöscht. Der Regler ist ausgeschaltet. |
| 3 | GESCHW | 0 | Stellungs-Algorithmus |
| | | 1 | Geschwindigkeits-Algorithmus |
| 4 | HANDART | 0 | **Bei GESCHW = 0:**<br>Nach dem Umschalten auf Handbetrieb wird der angegebene Stellwert YA in 4 Abtastschritten exponentiell auf den eingestellten Handwert geführt. Danach werden weitere Handwerte sofort am Reglerausgang übernommen.<br>**Bei GESCHW = 1:**<br>Die Handwerte werden sofort auf den Reglerausgang durchgeschaltet. Im Handbetrieb sind die Begrenzungen wirksam. |
| | | 1 | **Bei GESCHW =0:**<br>Die zuletzt ausgegebene Stellgröße wird beibehalten.<br>**Bei GESCHW = 1:**<br>Das Stellinkrement $dY_{,k}$ wird Null gesetzt. |
| 5 | NOZ | 0 | mit Störgrößenaufschaltung |
| | | 1 | keine Störgrößenaufschaltung |
| 6 bis 15 | - | | diese Bits werden vom PID-Algorithmus als Hilfsmerker verwendet. |

Das Reglerprogramm kann mit Festwerten oder Parametern versorgt werden. Die Eingabe der Parameter erfolgt über die zugeordneten Datenwörter. Dem Regler liegt ein PID-Algorithmus zugrunde. Sein Ausgangssignal kann wahlweise als Stellgröße (Stellungs-Algorithmus) oder als Stellgrößenänderung (Geschwindigkeits-Algorithmus) ausgegeben werden.

**Geschwindigkeits-Algorithmus**

Zu einem bestimmten Zeitpunkt t = k • TA wird das jeweilige Stellinkrement $dY_k$ nach folgender Formel berechnet:

- ohne Störgrößenaufschaltung (D11.5 = 1) und XW-Zuführung an Differenzierer (D11.1 =0)

$dY_k = K[(XW_k-XW_{k-1})R+TI \bullet XW_k+1/2(TD(XW_k - 2XW_{k-1} + XW_{k-2}) + dD_{k-1})]$
$= K (dPW_k + dI_k + dD_k)$

- mit Störgrößenaufschaltung (D11.5 = 0) und XW-Zuführung an Differenzierer (D11.1 =0)

$dY_k = K[(XW_k-XW_{k-1})R+TI \bullet XW_k+1/2(TD(XW_k - 2XW_{k-1} + XW_{k-2}) + dD_{k-1})] + (Z_k-Z_{k-1})$
$= K (dPW_k + dI_k + dD_k) + dZ_k$

- ohne Störgrößenaufschaltung (D11.5 = 1) und XZ-Zuführung an Differenzierer (D11.1 =1)

$dY_k = K[(XW_k-XW_{k-1})R+TI \bullet XW_k+1/2(TD(XZ_k - 2XZ_{k-1} + XZ_{k-2}) + dD_{k-1})]$
$= K (dPW_k + dI_k + dD_k)$

- mit Störgrößenaufschaltung (D11.5 = 0) und XZ-Zuführung an Differenzierer (D11.1 =1)

$dY_k = K[(XW_k-XW_{k-1})R+TI \bullet XW_k+1/2(TD(XZ_k - 2XZ_{k-1} + XZ_{k-2}) + dD_{k-1})] + (Z_k-Z_{k-1})$
$= K (dPW_k + dI_k + dD_k) + dZ_k$

P-Anteil I-Anteil D-Anteil Z-Anteil k: k-te Abtastung

Bei XWk-Zuführung gilt:
$XW_k = W_k - X_k$
$PW_k = XW_k - XW_{k-1}$
$QW_k = PW_k - PW_{k-1}$
$= XW_k-2XW_{k-1} + XW_{k-2}$

Bei XZ-Zuführung gilt:
$PZ_k = XZ_k - XZ_{k-1}$
$QZ_k = PZ_k - PZ_{k-1}$
$= XZ_k - 2XZ_{k-1} + XZ_{k-2}$

Damit erhält man:
$dPW_k = (XW_k - XW_{k-1})R$
$dI_k = TI \bullet XW_k$
$dD_k = ½(TD \bullet QW_k + dD_{k-1})$ bei XW-Zuführung
$= ½(TD \bullet QZ_k + dD_{k-1})$ bei XZ-Zuführung

**Stellungs-Algorithmus**

Beim Stellungsalgorithmus wird der gleiche Rechenalgorithmus wie beim Geschwindigkeitsalgorithmus herangezogen.
Der Unterschied zum Geschwindigkeitsalgorithmus besteht darin, dass zum Abtastzeitpunkt $t_k$ nicht das zu diesem Zeitpunkt berechnete Stellinkrement $dY_k$, sondern die Summe aller bis dahin berechneten Stellinkremente ausgegeben wird (im DW 48).

Zum Zeitpunkt $t_k$ wird die Stellgröße $Y_k$ folgendermaßen berechnet:

$$Y_k = \sum_{m=0}^{m=k} dYm$$

**Parametrierung des PID-Algorithmus**

Die Schnittstelle des OB 251 zu seiner Umgebung ist der Regler-DB.

Alle zur Berechnung des nächsten Stellwertes nötigen Daten sind im Regler-DB abgelegt.

Jeder Regler benötigt einen eigenen Regler-DB.

Die reglerspezifischen Daten werden in diesem Regler-DB parametriert, der DB muss mindestens 49 Datenwörter umfassen.

Ist ein zu kurzer oder kein DB aufgeschlagen, so geht die CPU mit Transfer-Fehler (TRAF) in STOP.

**Hinweis!**

Vor Aufruf des Regelalgorithmus OB 251 muss der zugehörige Regler-DB aufgeschlagen werden!

**Aufbau des Regler-DBs**

| Daten-wort | Name | Bemerkungen |
|---|---|---|
| 1 | K | Proportionalbeiwert (-32768 bis +32767) bei Reglern ohne D-Anteil<br>Proportionalbeiwert (-1500 bis +1500) bei Reglern ohne D-Anteil *<br>Bei positivem Regelsinn ist K größer als Null, bei negativem Regelsinn kleiner als Null; der angegebene Wert wird mit dem Faktor 0,001 **multipliziert. |
| 3 | R | R-Parameter (-32768 bis +32767) bei Reglern ohne D-Anteil<br>R-Parameter (-1500 bis +1500) bei Reglern ohne D-Anteil *<br>Gewöhnlich gleich 1 bei Reglern mit P-Anteil; der angegebene Wert wird mit dem Faktor 0,001 ** multipliziert. |
| 5 | TI | Konstante TI (0 bis9999)<br>Der angegebene Wert wird mit dem Faktor 0,001 multipliziert. |
| 7 | TD | Konstante TD (0 bis999) |
| 9 | W | Sollwert (-2047 bis + 2047) |
| 11 | STEU | Steuerwort (Bitmuster) |
| 12 | YH | Wert für Handbetrieb (-2047 bis + 2047) |
| 14 | BGOG | Obere Begrenzung (-2047 bis + 2047) |
| 16 | BGUG | Untere Begrenzung (-2047 bis + 2047) |
| | | |
| 22 | X | Istwert (-2047 bis + 2047) |
| 24 | Z | Störgröße (-2047 bis + 2047) |
| | | |
| 29 | XZ | Zugeführter D-Anteil (-2047 bis + 2047) |
| | | |
| 48 | YA | Ausgangsgröße (-2047 bis + 2047) |
| | | |

* Größere Verstärkungen sind möglich, wenn sprunghafte Änderungen der Regeldifferenz genügend klein sind. Große Änderungen der Regeldifferenz sind deshalb in mehrere kleine Änderungen aufzuteilen; z.B. durch Zuführen des Sollwertes über eine Rampenfunktion.

** Der Faktor 0,001 ist ein Näherungswert. Der exakte Wert für den Faktor ist 1/1024 bzw. 0,000976.

Alle angegebenen Parameter (mit Ausnahme des Steuerwortes STEU) sind als 16 bit-Festpunktzahl vorzugeben.

**Hinweis!**

Die in der Tabelle nicht aufgeführten Datenwörter werden vom PID-Algorithmus als Hilfsmerker verwendet.

**Initialisierung und Aufruf**

**PID-Regler im SPS-Programm**

Durch mehrmaligen Aufruf des OB 251 können verschiedene PID-Regler realisiert werden.

Vor jedem Aufruf des OB 251 muss auch der zugehörige Regler-DB aufgeschlagen werden.

**Hinweis!**

Im High-Byte des Steuerwortes DW 11 (DL11) sind wichtige Reglerinformationen gespeichert. Deshalb ist bei der Beeinflussung der Anwenderbits im Steuerwort darauf zu achten, dass schreibend nur mit T DR11 bzw. SU D 11.0 bis 11.7 oder RU D 11.0 bis 11.7 zugegriffen wird.

**Wahl der Abtastzeit**

Um die bekannte analoge Betrachtungsweise auch bei digitalen Regelkreisen anwenden zu können, darf die Abtastzeit nicht zu groß gewählt werden.

Die Erfahrung hat gezeigt, dass eine Abtastzeit $T_A$ von etwa 1/10 der Zeitkonstante $T_{RKdom}$ (=dominierende Streckenzeitkonstante des geschlossenen Regelkreises) zu einem mit dem analogen Fall vergleichbaren Regelergebnis führt.

Die Zeitkonstante $T_{RKdom}$ bestimmt die Sprungantwort des geschlossenen Regelkreises.

$TA = 1/10 \bullet T_{RKdom}$

Um eine konstante Abtastzeit zu gewährleisten, ist der OB 251 grundsätzlich im Zeit-OB (OB 13) aufzurufen.

**Beispiel** **Aufruf eines Reglers**

| **OB 13** | | | | | **Erläuterung** |
|---|---|---|---|---|---|
| | : | | | | |
| | : | SPA | FB | 10 | Regler bearbeiten |
| NAME | : | REGLER 1 | | | Die Reglerabtastzeit wird durch die OB 13 - Aufrufzeit bestimmt (Einstellung im SD 97). Bei der Wahl der Abtastzeit muss die Verschlüsselungszeit der verwendeten Analogeingabebaugruppen beachtet werden. |
| | : | BE | | | |

| **FB 10** | | | | | **Erläuterung** |
|---|---|---|---|---|---|
| | : | | | | |
| NAME | : | Regler 1 | | | |
| | : | | | | |
| | : | A | DB | 30 | Regler - DB aufschlagen |
| | : | | | | |
| | : | | | | **Steuerbits für Regler einlesen** |
| | : | L | PB | 0 | Steuereingänge für Regler |
| | : | T | MB | 10 | einlesen |
| | : | T | DR | 11 | und in DR 11 abspeichern |
| | | | | | **Achtung:** In DL 11 sind wichtige Steuerinformationen für den OB 251 gespeichert, deshalb müssen die Steuerbits mit T DR 11 übertragen werden um DL 11 nicht zu beeinflussen |
| | | | | | **Ist- und Sollwert einlesen** |
| | : | U | M | 12.0 | Nullmerker (für nicht verwendete Funktionen im FB 250) |
| | : | R | M | 12.0 | |
| | : | UN | M | 12.1 | 1-Merker |
| | : | S | M | | |
| | : | | | | |
| | : | SPA | FB | 250 | Istwert einlesen |
| NAME | : | RLG:AE | | | Integrierter Standard FB (VIPA) |
| BG | : | | KF | *128 | Baugruppenadresse |
| KNKT | : | | KY | 0,6 | Kanalnrummer 0, Festpunkt bipolar |
| OGR | : | | | +2047 | Obergrenze Istwert |
| UGR | : | | | -2047 | Untergrenze Istwert |
| EINZ | : | | M | 12.0 | Keine Einzelabtastung |
| XA | : | | DW | 22 | Norm. Istwert in Regler-DB ablegen |
| FB | : | | M | 12.2 | Fehlerbit |
| BU | : | | M | 12.3 | Bereichsüberschreitung |
| TBIT | : | | M | 12.4 | Tätigkeitsbit |
| | : | | | | |
| | : | | | | |

**Beispiel-Fortsetzung**

| **FB 10** | | | | | **Erläuterung** |
|---|---|---|---|---|---|
| Fort-setzung | | | | | |
| | : | SPA | FB | 250 | Sollwert einlesen |
| NAME | : | RLG:AE | | | |
| BG | : | | KF | +128 | Baugruppenadresse |
| KNKT | : | | KY | 1,6 | Kanalnrummer 1, Festpunkt bipolar |
| OGR | : | | | +2047 | Obergrenze Istwert |
| UGR | : | | | -2047 | Untergrenze Istwert |
| EINZ | : | | M | 12.0 | Keine Einzelabtastung |
| XA | : | | DW | 9 | Norm. Istwert in Regler-DB ablegen |
| FB | : | | M | 13.1 | Fehlerbit |
| BU | : | | M | 13.2 | Bereichsüberschreitung |
| TBIT | : | | M | 13.3 | Tätigkeitsbit |
| | : | | | | |
| | : | | | | |
| | : | U | M | 10.0 | Im Handbetrieb wird Sollwert gleich dem |
| | : | SPB | | =WEIT | Istwert gesetzt; damit Regler auf eine |
| | : | L | DW | 22 | eventuell vorhandene Regelabweichung mit |
| | : | T | DW | 9 | einem P-Sprung reagiert, wenn in den |
| | : | | | | AUTOMATIK-Betrieb umgeschaltet wird |
| =WEIT | : | | | | |
| | : | | | | |
| | : | | | | |
| | : | SPA | OB | 251 | **Regler-Aufruf** |
| | : | | | | |
| | : | | | | |
| | : | | | | |
| | : | | | | **Stellwert Y ausgeben** |
| | : | | | | |
| | : | SPA | FB | 251 | Integrierter Standard-FB (VIPA) |
| NAME | : | RLG:AA | | | |
| XE | : | | DW | 48 | Stellgröße Y an Analogausgabe |
| BG | : | | KF | +176 | Baugruppenadresse |
| KNKT | : | | KY | 0,1 | Kanal 0, Festpunkt bipolar |
| OGR | : | | KF | +2047 | Obergrenze des Stellsignals |
| UGR | : | | KF | -2047 | Untergrenze des Stellsignals |
| FEH | : | | M | 13.5 | Fehlerbit bei Grenzwertvorgabe |
| BU | : | | M | 13.6 | Bereichsüberschreitung |
| | : | | | | |
| | : | | | | |
| | : | BE | | | |

**Beispiel** **Regler - Datenbaustein DB 30**

| **DB 30** | | | | **Erläuterung** |
|---|---|---|---|---|
| 0: | KH | = | 0000; | |
| 1: | KF | = | +01000; | K-Parameter (hier = 1), Faktor 0.001 |
| 2: | KH | = | 0000; | (Wertebereich: -32768 bis +32767) |
| 3: | KF | = | +01000; | R-Parameter (hier = 1), Faktor 0.001 |
| 4: | KH | = | 0000; | (Wertebereich: -32768 bis +32767) |
| 5: | KF | = | +00010; | TI=TA/TN (hier = 0.01), Faktor 0.001 |
| 6: | KH | = | 0000; | (Wertebereich: 0 bis 9999) |
| 7: | KF | = | +00010; | TD=TV/TA (hier = 10), Faktor 1 |
| 8: | KH | = | 0000; | (Wertebereich: 0 bis 999) |
| 9: | KF | = | +00000; | Sollwert W, Faktor 1 |
| 10: | KH | = | 0000; | (Wertebereich: -2047 bis +2047) |
| 11: | KM | = | 00000000 00100000 | Steuerwort |
| 12: | KF | = | +00500 | Handwert YH, Faktor 1 |
| 13: | KH | = | 0000; | (Wertebereich: -2047 bis +2047) |
| 14: | KF | = | +02000; | Obere Regl.Begr. BGOG, Faktor 1 |
| 15: | KH | = | 0000; | (Wertebereich: -2047 bis +2047) |
| 16: | KF | = | -02000; | Untere Regl.Begr. BGUG, Faktor 1 |
| 17: | KH | = | 0000; | (Wertebereich: -2047 bis +2047) |
| 18: | KH | = | 0000; | |
| 19: | KH | = | 0000; | |
| 20: | KH | = | 0000; | |
| 21: | KH | = | 0000; | |
| 22: | KF | = | +00000; | Istwert X, Faktor 1 |
| 23: | KH | = | 0000; | (Wertebereich: -2047 bis +2047) |
| 24: | KF | = | +00000; | Stoergröße, Faktor 1 |
| 25: | KH | = | 0000; | (Wertebereich: -2047 bis +2047) |
| 26: | KH | = | 0000; | |
| 27: | KH | = | 0000; | |
| 28: | KH | = | 0000; | |
| 29: | KF | = | +00000; | XZ-Aufschaltung für Diff., |
| 30: | KH | = | 0000; | Faktor 1, (-2047 bis + 2047) |
| 31: | KH | = | 0000; | |
| 32: | KH | = | 0000; | |
| 33: | KH | = | 0000; | |
| 34: | KH | = | 0000; | |
| 35: | KH | = | 0000; | |
| 36: | KH | = | 0000; | |
| 37: | KH | = | 0000; | |
| 38: | KH | = | 0000; | |
| 39: | KH | = | 0000; | |
| 40: | KH | = | 0000; | |
| 41: | KH | = | 0000; | |
| 42: | KH | = | 0000; | |
| 43: | KH | = | 0000; | |
| 44: | KH | = | 0000; | |
| 45: | KH | = | 0000; | |
| 46: | KH | = | 0000; | |
| 47: | KH | = | 0000; | |
| 48: | KF | = | +00000; | Reglerausgang Y, Faktor 1 |
| 49: | KH | = | 0000; | (Wertebereich: -2047 bis +2047) |
| 50 | | | | |

# Integrierter Baustein DB 1

**Übersicht**

Die CPU 24x verfügt über Funktionen, die Sie nach Ihrem Bedarf einstellen (parametrieren) können, wie:

- Manuelle Adressvorgabe bei direkt gesteckten Modulen,
- Parametrierung direkt gesteckter Module,
- Datenbereiche in CPU 24x DP zuweisen,
- Aufrufintervalle für zeitgesteuerte Programmbearbeitung,
- Systemeigenschaften einstellen.

**Hinweis!**

System-200V-spezifische Einstellungen können Sie auch unter **WinNCS** von VIPA programmieren. Hierbei wird ein DB1 erzeugt, den Sie als s5d-Datei exportieren und in die CPU übertragen können.

**Aufbau und Voreinstellung des DB 1**

Zur Erleichterung der Parametrierung ist ein DB 1 mit voreingestellten Werten (Default-Parametern) bereits in der CPU integriert. Wenn Sie den Default-DB1 nach "Urlöschen" vom AG ins PG laden und sich am Bildschirm anzeigen lassen, hat er folgenden Aufbau:

```
Baustein #DB 1
BIB #0
00000:           KC = DB1¨TFB:¨OB13¨10¨;¨SDP:¨;
00012            KC = ¨WD¨500¨;¨END¨;
```

Leerzeichen sind hier durch ¨ dargestellt.

Dieser voreingestellte DB enthält für jede Funktion einen Parameterblock. Jeder Parameterblock beginnt mit einer Blockkennung, gefolgt von einem Doppelpunkt. Innerhalb der Parameterblöcke sind die einzelnen Parameter für die jeweiligen Funktionen zusammengefasst.

Ein Parameterblock beginnt immer mit einer Blockkennung, gefolgt von einem Doppelpunkt. Hinter dem Doppelpunkt muss mindestens ein Leerzeichen stehen. Der Strichpunkt (;) kennzeichnet das Ende eines Parameterblocks.

**Parameterblöcke**

Für die CPU 24x von VIPA existieren folgende Parameterblöcke:

| Blockkennung | Bedeutung/Voreinstellung |
|---|---|
| ′DB1′ | Anfangskennung |
| ′SL1: ′; | Aus Kompatibilitätsgründen ist SL1 vorhanden. Es wird aber nichts ausgeführt. |
| ′CLP: ′; | **Cl**ock-**P**arameters: Parameterblock für integrierte Uhr/ keine Uhrenfunktion aktiviert. |
| ′UAT: ′; | **U**ser **A**dress **T**able<br>Manuelle Adressvorgabe für direkt gesteckte Module |
| ′Pxx: ′; | Parametrierung direkt gesteckter Module. |
| ′DPS: ′; | **DP S**lave<br>Datenbereich für Profibus-Koppler in CPU 24x DP neu vergeben. |
| ′SDP: ′; | **S**ystem-**D**ependent-**P**arameter:Parameterblock für Systemeigenschaften/ Zykluszeitüberwachung ist auf 500 ms voreingestellt. |
| ′TFB: ′; | **T**imer**-F**unktions-**B**austeine: Parameterblock für zeitgesteuerte Programmbearbeitung: OB 13 wird alle 100 ms aufgerufen. |
| ′ERT: ′; | **E**rror **R**eturn: Adresse für Parametrierfehler-Code/keine Default-Einstellung. |
| ′END ′; | **End**ekennung des DB 1. |

Die Reihenfolge der Parameterblöcke im DB 1 ist nicht festgelegt; einzelne Blöcke sind mit dem Strichpunkt (;) voneinander zu trennen. Zwischen Strichpunkt und der nächsten Blockkennung ist mindestens ein Leerzeichen einzufügen.

**Parametrierfehler-Code**

**Im DB 1 die Adresse für den Parametrierfehler-Code festlegen**

Der Parameterblock "ERT:" bietet eine komfortable Möglichkeit, Parametrierfehler zu beheben. Darum sollten Sie diesen Block im DB 1 ergänzen, bevor Sie andere Parameter verändern oder einfügen. Da der Parameterblock nur während der Inbetriebnahmephase von Bedeutung ist, sollten Sie ihn aus Platzgründen im "normalen" Betrieb löschen.

Um Parametrierfehler leichter zu finden und besser beheben zu können, können Sie sich vom AG Fehlermeldungen in codierter Form ausgeben lassen. Dazu müssen Sie der CPU nur angeben, wo sie den Fehlercode ablegen soll. Diese Angabe machen Sie im Parameterblock "ERT:" des DB 1.

Der Fehlercode kann abgelegt werden in:

- Merkerworten,
- Datenworten eines Datenbausteins.

Der gesamte Fehlercode belegt 20 Merkerbytes bzw. 10 Datenworte. Im Parameterblock "ERT:" geben Sie lediglich die Anfangsadresse für den Fehlercode an. Wie z.B.:

ERT: ERR MBx

oder

ERT: ERR DBxDWy

**Vorgehen beim Parametrieren des DB 1**

- Default-DB1 am PG ausgeben lassen,
- mit Cursor in den gewünschten Parameterblock springen,
- Parameter ändern/ergänzen,
- geänderten DB 1 ins AG übertragen,
- CPU von STOP→RUN schalten.

Geänderter DB1-Parameter werden übernommen.

**Hinweis!**
Erkennt die CPU einen Parametrierfehler im DB 1, so bleibt die CPU auch nach dem Umschalten von STOP→RUN in STOP (Rote LED leuchtet).
Es wird außerdem das Bit "SYSFE" und das Sammelfehler Bit (BS7 Bit 2) im U-STACK gesetzt.

**Regeln für die Parametrierung des DB 1**

Der DB 1 besteht aus:

| | |
|---|---|
| • einer Anfangskennung | DB 1 |
| • ein bis mehreren Parameterblöcken | z.B.: SDP: WD 500 |

ein Parameterblock besteht aus:

| | |
|---|---|
| • einer Blockkennung | z.B.: SDP: |
| • ein bis mehreren Parametern | z.B.: WD 500 |

ein Parameter besteht aus:

| | |
|---|---|
| • einem Parameternamen | z.B.: WD |
| • ein bis mehreren Argumenten | z.B.: 500 |
| • einer Blockendekennung: | ; (Strichpunkt) |
| • einer Endekennung: | END |

**Beispiel**

```
DB1
KC=DB1"ERT:"ERR"MB200";;"TFB
KC=:"OB13"10";;"SDP:"WD"500";
KC=;;"END";
```

**Regeln**

Im Folgenden geben wir Ihnen sämtliche Regeln an, die Sie beachten müssen, wenn Sie im DB 1 Parameter ändern möchten oder ganze Parameterblöcke ergänzen wollen.

Diese Regeln sind einzuhalten, da sonst die CPU Ihre Eingaben nicht "verstehen" kann.

- Der DB 1 muss mit der Eingabe "DB1" beginnen. Die drei Zeichen dürfen nicht durch Füllzeichen voneinander getrennt sein. Hinter der Anfangskennung muss mindestens ein Füllzeichen folgen.
- Nach Anfangskennung incl. Füllzeichen folgt die Blockkennung eines Parameterblocks. Die Reihenfolge der Parameterblöcke im DB 1 ist beliebig. Die Blockkennung kennzeichnet einen Block zusammengehöriger Parameter. Unmittelbar hinter der Blockkennung muss ein Doppelpunkt (:) folgen. Wenn der Doppelpunkt fehlt, dann überspringt die CPU diesen Block und gibt eine Fehlermeldung aus. Hinter der mit einem Doppelpunkt abgeschlossenen Blockkennung muss mindestens ein Füllzeichen eingefügt sein.
- Es folgt ein Parametername. Parameternamen sind Namen für einzelne Parameter innerhalb eines Parameterblocks. Innerhalb eines Blocks müssen sich die ersten vier Zeichen eines Parameternamens voneinander unterscheiden. Hinter dem Parameternamen muss mindestens ein Füllzeichen eingefügt sein.
- Zu jedem Parametername gehört mindestens ein Argument. Bei einem Argument handelt es sich entweder um eine Zahl oder um einen Operanden, den Sie eingeben. Wenn mehrere Argumente zu einem Parameternamen gehören, dann müssen alle Argumente durch mindestens ein Füllzeichen voneinander getrennt sein. Dem letzten Argument muss (wiederum) mindestens ein Füllzeichen folgen.
- Das Blockende muss durch einen Strichpunkt (;) gekennzeichnet sein. Hinter dem Strichpunkt muss mindestens ein Füllzeichen eingegeben werden. Wenn Sie den Strichpunkt weglassen, dann führt das zu Fehlinterpretationen in der CPU.
- Danach können weitere Parameterblöcke folgen.
- Nach dem Ende des letzten Parameterblocks muss die Endekennung "END" eingegeben werden. Sie kennzeichnet das Ende des DB 1. Wenn Sie vergessen, diese Endekennung einzugeben, dann führt das zu Fehlern in der CPU.

*Als Füllzeichen sind generell erlaubt: Leerzeichen und Komma.*

**Kommentar**

Sie können Ihren DB 1 mit Kommentaren ergänzen.

Kommentare können Sie überall dort einfügen, wo auch ein Füllzeichen stehen darf. Das Kommentarzeichen (#) muss am Anfang und am Ende eines Kommentars stehen. Der Text zwischen zwei Kommentarzeichen darf kein weiteres # enthalten.

Dem Kommentar muss mindestens ein Füllzeichen folgen.

Zur Erleichterung der Lesbarkeit der Parameternamen, kann man beliebig viele Zeichen ergänzen, wenn man hinter der Kurzbezeichnung der Parameternamen einen Unterstrich anfügt.

z.B.: aus WD wird WD_Zykueberw.

Am Ende des ergänzten Parameternamens muss mindestens ein Füllzeichen folgen.

Als kleine Hilfe zur Kontrolle Ihres DB 1 kann folgende Faustregel dienen:

*Mindestens 1 Leerzeichen muss stehen:*

*nach der Anfangskennung und*

*vor und nach Blockkennung, Parametername, Argument und Strichpunkt.*

**Parametrierfehler erkennen und beseitigen**

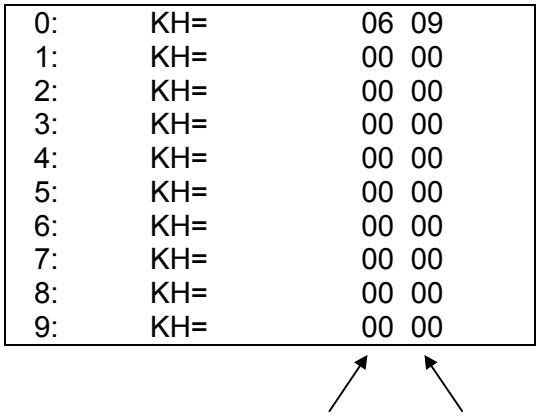
Sollte Ihnen dennoch einmal bei der Parametrierung ein Fehler unterlaufen sein und das AG nicht in den RUN-Zustand übergehen, dann haben Sie die Möglichkeit Parametrierfehler zu erkennen:

- mit Hilfe des Parametrierfehler-Codes,
- durch die Analysefunktion "USTACK".

**Beispiel**

Sie haben im Parameterblock "ERT:" die Anfangsadresse DB 3 DW0 angegeben und der so parametrierte DB 1 wurde bereits vom AG übernommen. Anschließend setzen Sie die Parametrierung des DB 1 fort. Nach Übertragung der geänderten DB1-Parameter ins AG stellen Sie fest, dass die CPU in STOP bleibt. Sie stellen als STOP-Ursache einen Parametrierfehler im USTACK fest. Um den Fehler zu finden, lassen Sie sich am PG den DB 3 ausgeben. Auf dem Bildschirm erscheint der gesamte Inhalt des DB 3; die Datenworte DW 0 bis DW 9 enthalten den Parametrierfehler-Code.

**Parametrierfehler-Codes** Bildschirmanzeige mit Parametrierfehler-Codes




| Fehlerursache (welcher Fehler ist aufgetreten) | DWL | DWR | Fehlerort (in welchem Parameterblock ist der Fehler aufgetreten) |
|---|---|---|---|
| kein Fehler | 00 | 00 | Fehler ist keinem Block zuzuordnen |
| Anfangs- oder Endkennung fehlt | 01 | 01 | Fehler ist keinem Block zuzuordnen |
| nicht abgeschl. Kommentar | 02 | 02 | Fehler ist keinem Block zuzuordnen |
| Strichpunkt vor END fehlt | | 03 | |
| Blockkennung Syntaxfehler | 03 | | |
| Parameter Syntaxfehler | 04 | 06 | CLP: Clock-Parameter (Uhr) |
| Argument Syntaxfehler | 05 | | |
| Bereichsüber- oder -unterschreitung | 06 | | |
| in einem Argument | | 09 | TFB: Timer-Funktions-Baustein |
| Parameterkombination nicht erlaubt | 07 | | |
| | | 11 | SDP: Systemdaten-Parameter |
| nicht definiert | 08 | | |
| nicht definiert | 09 | | |
| DB nicht vorhanden | 10 | | |
| Platz im DB reicht nicht aus | 11 | 99 | ERT: Error Return |
| Fehler bei der Wochentagsangabe | 12 | F0 | Fehler ist keinem Block zuzuordnen |
| Fehler im Datum | 13 | . | Fehler ist keinem Block zuzuordnen |
| Fehler in der Zeitangabe | 14 | . | Fehler ist keinem Block zuzuordnen |
| Fehler im Zeitformat | 15 | FF | Fehler ist keinem Block zuzuordnen |
| 24h/12 h-Modus) | | | |

**Parametrierfehler im "USTACK" lokalisieren**

Wenn die CPU im Anlauf einen Parametrierfehler im DB 1 feststellt, bleibt sie im STOP-Zustand und hinterlegt im USTACK wo der Fehler aufgetreten ist.

Der USTACK enthält sowohl die absolute (Fehler-) Adresse wie auch die relative (Fehler-) Adresse. Der STEP-Adresszähler (SAZ) im USTACK zeigt dann entweder **auf** die Adresse oder direkt **vor** die Adresse, die die fehlerhafte Eingabe enthält.

Hierbei handelt es sich um Byte-Adressen.

Außerdem werden im U-STACK die Bits SYSFE und das Sammelfehler Bit (BS7 Bit 2) gesetzt.

**Übernahme der DB 1-Parameter ins AG**

Anders als die anderen Datenbausteine, wird der DB 1 nur ein einziges Mal bearbeitet; und zwar bei einem Neustart des AG. Das wurde so eingerichtet, um den DB 1 für ganz bestimmte Sonderfunktionen auszurüsten.

Eine solche Sonderfunktion ist das Parametrieren des AGs mit Hilfe des DB 1. Parametrieren heißt, Sie geben im Datenbaustein DB 1 die Parameter für diejenigen internen Funktionen ein, mit denen ihr AG arbeiten soll.

Diese Eingaben im DB 1 werden nur bei einem Neustart in das Betriebssystem des AGs übernommen. Jede Änderung im DB 1 muss deshalb ein entsprechender Neustart folgen, den Sie veranlassen durch das Umschalten von Netz-Aus→Netz-Ein oder STOP→RUN.

Das AG übernimmt die Parameter des DB 1 und hinterlegt sie im Systemdatenbereich.

**Hinweis!**

Die CPU bleibt in STOP, wenn sie im Anlauf einen Parametrierfehler feststellt. Im Bedienfeld leuchtet dann die rote LED und im USTACK wird eine Fehleradresse des DB 1 angegeben.

**DB 1-**
**Parametrierung**

| Parameter | Argument | Bedeutung |
|---|---|---|
| Blockkennung: SL1: | | SINEC L1 von Siemens (aus Kompatibilitätsgründen vorhanden - wird nicht ausgewertet) |
| SLN | p | Slave-Nummer |
| SF<br>EF<br>KBE<br>KBS<br>PGN | DBxDWy<br>DBxDWy<br>MBy<br>MBy<br>p | Lage des Sendefachs<br>Lage des Empfangsfachs<br>Lage des Koordinierungsbytes "Empfangen"<br>Lage des Koordinierungsbytes "Senden"<br>PG-Busnummer |
| p = 1...30 x = 2...255 y = 0...255 | | |
| Blockkennung: SDP: | | System-Dependent-Parameter |
| WD | p | Zykluszeitüberwachung (Watch-Dog-Timer) |
| p = 1...2550 | | |
| Blockkennung: UAT: | | User Adress Table |
| | | |
| Blockkennung: Pxx: | | Parametrierung direkt gesteckter Module |
| | | |
| Blockkennung: DPS: | | DP-Slave Datenbereich für Profibus-Koppler in CPU 24xDP neu vergeben |
| | | |
| Blockkennung: TFB: | | Timer-Funktions-Baustein |
| OB 10<br>OB 11<br>OB 12<br>OB 13 | p | Intervall (10 ms), in dem der OB 13 aufgerufen und bearbeitet wird (OB 13, 50 entspricht 500ms). |
| p = 0..65535 (in 10 ms-Schritten angeben) | | |
| Blockkennung: ERT: | | Error Return table |
| ERR | MBx<br>DBxDWy | Error Return |
| | | |

*Fortsetzung ...*

*... Fortsetzung*

| Parameter | Argument | Bedeutung |
|---|---|---|
| Blockkennung: CLP: | | **Cl**ock-**P**arameters (Uhr) |
| CF | p | Korrekturfaktor eingeben (**C**orrection **F**actor) |
| CLK | DBxDWy,<br>MWz,EWv<br>oder AWv | Lage der Uhrendaten (**Cl**oc**k** Data) |
| STW | DBxDWy,<br>MWz,EWv<br>oder AWv | Lage des Statuswortes (**St**atus **W**ord) |
| STP | J/Y/N | Uhr im STOP-Zustand aktualisieren (**ST**O**P**) |
| SAV | J/Y/N | Uhrzeit nach letztem STOP→RUN-Übergang bzw. NETZ AUS retten (**SAV**e) |
| OHE | J/Y/N | Betriebsstundenzähler freigeben<br>(**O**peration **H**our counter **E**nable) |
| SET | wt tt.mm.jj<br>hh:mn:ss[1]<br>AM/PM[2] | Uhrzeit/Datum stellen |
| TIS | wt tt.mm.<br>hh:mn:ss[1]<br>AM/PM[2] | Weckzeit stellen (**T**imer **I**nterrupt **S**et) |
| OHS | hhhhhh:mn.ss[1] | Betriebsstundenzähler stellen<br>(**O**peration **H**our counter **S**et) |
| wt = 1...7 (Wochentag=So...Sa)<br>tt = 01...31 (Tag)<br>mm = 01...12 (Monat)<br>jj = 0...99 (Jahr)<br>hh = 00...23 (Stunden)<br>mn = 00...59 (Minuten)<br>ss = 00...59 (Sekunden)<br>hhhhhh = 0...999999 (Stunden) | | p = -400...400<br>v = 0...126<br>x = 2...255<br>y = 0...255<br>z = 0...254<br>j/J = ja<br>y/Y = ja<br>n/N = nein |

1) Soll ein Argument (z.B. Sekunden) nicht übernommen werden: XX eingeben! Die Uhr läuft mit den aktuellen Daten weiter. Im Parameterblock TIS wird dieses Argument nicht berücksichtigt.

2) Geben Sie AM oder PM nach der Uhrzeit an, läuft die Uhr im 12 Stunden-Modus. Lassen Sie diese Argument weg, läuft die Uhr im 24 Stunden-Modus. In den Parameterblöcken SET und TIS muss der gleiche Zeitmodus verwendet werden.

**DB1-Programm-beispiel**

Das folgende Beispiel eines DB1-Programms stellt Ihnen die komplette DB1-Parametrierung noch einmal vor.

Es wurden parametriert:

- die Systemeigenschaften,
- die zeitgesteuerte Bearbeitung,
- die Platzierung von FB,
- die integrierte Uhr,
- die Adresse für Parametrierfehler-Code.

| | AWL | Erläuterung |
|---|---|---|
| 0: | KC='DB1 | DB1-Kopfkennung |
| 12: | KC='#Systemeigenschaften# | Kommentar |
| 24: | KC='SDP: WD_Zykueberw 500 | Blockkenn. u. Param. Zykl-Überw. |
| 36: | KC='RDLY_Anlaufverz 1000 | Anlaufverzögerung |
| 48: | KC='RT_Remanenz_Timer n | Remanenz der Zeiten (halb oder ganz) |
| 60: | KC='RC_Remanenz_Zaehler n | Remanenz der Zähler (halb oder ganz) |
| 72: | KC='RF_Remanenz_Merker n | Remanenz der Merker (halb oder ganz) |
| 84: | KC='PROT_Softwareschutz n | Softwareschutz aktiv oder nicht |
| 96: | KC='PIO_PAA_Sperre n | Prozessabbild d. Ausgänge sperren |
| 108: | KC='PII_PAE_Sperre n | Prozessabbild d. Eingänge sperren |
| 120: | KC='PRIO_OB6_Priorität 0 | Priorität des OB 6 gegenüber OB 2 |
| 132: | KC='; | Blockendekennung |
| : | | |
| 240: | KC='#Zeitgest. Bearb.# | Kommentar |
| 252: | KC='TFB: | Blockkennung für zeitgest. Bearb. |
| 264: | KC='OB10_Intervall 400 | Aufrufintervall OB10 |
| 276 | KC='OB11_Intervall 300 | Aufrufintervall OB11 |
| 288: | KC='OB12_Intervall 200 | Aufrufintervall OB12 |
| 300: | KC='OB13_Intervall 100 | Aufrufintervall OB13 |
| 312 | KC='; | Blockendekennung |
| : | | |
| 324: | KC='#Plazierung int. FB# | Kommentar |
| 336: | KC='PFB: | Blockkennung Plaz. FB238/239 |
| 348: | KC='SFB_Compr 238 210 | FB238 bekommt die Nummer 210 |
| 360: | KC='SFB_Delete 239 211 | FB239 bekommt die Nummer 211 |
| 372: | KC='; | Blockendekennung |
| : | | |
| 564: | KC='# Fehlercodebereich # | Kommentar |
| 576: | KC='ERT:_ | Blockkennung Fehlercodebereich |
| 588: | KC='ERR_Fehlercode DB4DW0 | Fehlercodebereich |
| 600: | KC='; | |
| 612: | KC='END | DB1-Endekennung |

# Teil 11 Uhrfunktionen

**Überblick**

In diesem Kapitel ist der Aufbau und die Parametrierung der integrierten Uhr beschrieben. Eine integrierte Uhr besitzen die CPU-Typen: CPU 243 und CPU 244.

**Inhalt**


| Thema | Seite |
|---|---|
| **Teil 11 Uhrfunktionen** | **11-1** |
| Allgemeines | 11-2 |
| Parametrierung der integrierten Uhr | 11-3 |
| Aufbau des Uhrdatenbereichs | 11-7 |
| Aufbau des Statusworts | 11-10 |
| Uhr stellen und lesen | 11-12 |
| Weckfunktionen | 11-19 |
| Betriebsstundenzähler | 11-22 |

# Allgemeines

Die CPU-Typen **CPU 243** und **CPU 244** besitzen eine integrierte Uhr.

Über eine integrierte Uhr haben Sie weitere Möglichkeiten zur Kontrolle eines Prozessablaufs:

- Weck- und Alarmfunktionen
  z.B. zur Überwachung der Zeitdauer eines Prozesses.
- Betriebsstundenzähler
  z.B. zur Überwachung der Inspektionsintervalle.
- Uhrfunktion
  z.B. zur Feststellung des Zeitpunktes, an dem die CPU in einem Fehlerfall in den STOP-Zustand überging.

Die Genauigkeit der Uhr beträgt bei 25°C ± 22 Sekunden (maximal) pro Tag.

**Pufferung der Hardwareuhr**

Bei vorhandener Batterie läuft die Uhr auch bei NETZ-AUS weiter. Ist das AG nicht mit einer Batterie gepuffert, werden nach NETZ-EIN bei der Uhreninitialisierung die Uhrenwerte auf 01.01.89 12.00.00 Uhr, Wochentag: 1 gestellt. Ebenfalls voreingestellt ist der 24-h-Modus.

# Parametrierung der integrierten Uhr

**Voraussetzungen**

Für den Einsatz der Uhr ist ein Uhrdatenbereich und ein Statuswort erforderlich.

Hierzu müssen folgende Informationen in den Systemdaten 8 bis 10 hinterlegt werden:

- Die Lage des Uhrdatenbereiches,
- Die Lage des Statuswortes.

**Parametrierung**

Sie haben die Möglichkeit, die Uhr über den DB 1 zu parametrieren (Parameterblock CLP).

Die andere Möglichkeit, die integrierte Uhr zu parametrieren, ist im Folgenden erläutert.

In einem von Ihnen zu programmierenden FB, der zweckmäßigerweise, durch einen der beiden Anlauf-OBs OB 21 und OB 22 aufgerufen wird, können Sie die Funktion der Uhr parametrieren.

Im Funktionsbaustein werden die Parameter mit Transferoperationen (z.B. "TBS, TNB") im entsprechenden Systemdatum abgelegt.

Für die Lage des Uhrdatenbereiches und des Statuswortes sind die Systemdatenworte 8 bis 10 verantwortlich. Hier wird festgelegt, ob es sich um einen Merkerbereich oder um einen Datenbaustein handelt. Des weiteren wird die genaue Lage innerhalb des definierten Bereiches bestimmt. Das Betriebssystem nimmt keine Standardbelegung dieser Systemdatenzellen vor, so dass im Standardfall kein Zugriff auf die Uhr möglich ist.

Die folgende Tabelle gibt Aufschluss über die Bedeutung der einzelnen Bytes der Systemdatenworte 8 bis 10. Die Systemdatenworte 11 und 12 werden im Anschluss an die Tabelle erläutert.

Systemdatenworte für Uhr

| absolute Adresse RAM-Speicher | | Systemdatenwort BS | Bedeutung | zulässige Parameter |
|---|---|---|---|---|
| CPU243 | CPU244 | | | |
| EA10 | 1DA10 | 8 | Operandenbereich der Uhrdaten | ASCII-Zeichen: D für DB-Bereich<br>M für Merkerbereich |
| EA11 | 1DA11 | 8 | Anfangsadresse Uhrdaten<br>Operandenbereich D<br>Operandenbereich M | Nummer des DB<br>(DB 2...DB 255) oder<br>Merkerbyte-Nummer |
| EA12 | 1DA12 | 9 | Anfangsadresse Uhrdaten (nur relevant für Operandenbereich D) | Nummer des Datenwortes<br>DW 0...DW 255 |
| EA13 | 1DA13 | 9 | Operandenbereich des Statuswortes | ASCII Zeichen:<br>D für DB-Bereich<br>M für Merkerbereich |
| EA14 | 1DA14 | 10 | Adresse des Statuswortes<br>Operandenbereich D<br>Operandenbereiche M | Nummer des DB<br>(DB 2...DB 255) oder<br>Merkerwort-Nummer |
| EA15 | 1DA15 | 10 | Adresse des Statuswortes (nur relevant für Operandenbereich D) | Nummer des Datenwortes<br>DW 0 ...DW 255 |
| EA16 | 1DA16 | 11 | Anlaufkontrolle des Uhrbausteins | |
| EA17 | 1DA17 | 11 | Anlaufkontrolle des Uhrbausteins | |
| EA18 | 1DA18 | 12 | Korrekturwert* | -400...0...+400 |
| EA19 | 1DA19 | 12 | Korrekturwert* | -400...0...+400 |

* wird nur einmal pro Stunde geprüft und verarbeitet

**BS 11 Initialisierung der Uhr**

Während der Initialisierung des Uhrbausteins prüft das System die Ansprechbarkeit durch das Betriebssystem und den Anlauf des Uhrenchips.

Zu diesem Zweck stehen im Systemdatum BS 11 die Bits 0 und 1 zur Verfügung. Über die Systemanweisung L BS 11 im Anwenderprogramm können Sie den Uhrenstatus ermitteln.

| Systemdatum 11 (EA16h) | | Bedeutung |
|---|---|---|
| Bit 1 | Bit 0 | |
| 0 | 0 | Uhrchip nicht vorhanden |
| 0 | 1 | Uhrchip nicht ansprechbar (defekt) |
| 1 | 0 | Uhrchip läuft nicht an |
| 1 | 1 | Uhrchip läuft ordnungsgemäß |

Das Systemdatum 11 ist bereits in den Anlauf-OBs 21 und 22 abfragbar, d.h. Sie können gegebenenfalls den Nichtanlauf der Uhr feststellen und eine Meldung ausgeben.

*Beispiel*

Initialisierung der Uhr im ANLAUF des AGs (OB 21 und OB 22).

Die Uhrdaten sollen im Datenbaustein 2 ab Datenwort 0 abgelegt werden. Das Statuswort wird im Merkerwort 10 abgelegt. Merker 12.0 ist gesetzt, wenn die Uhr nicht ordnungsgemäß angelaufen ist.

OB 21

| OB 21 AWL | Erläuterung |
|---|---|
| :SPA FB 101 | Uhr initialisieren |
| NAME:UHR-INIT | |
| TUDA: KC DB | Uhrdatenbereich liegt im DB. |
| NUDA: KY 2,0 | hier : DB2 ab DW 0 |
| TUSW: KC MW | Satuswort der Uhr ist MW |
| NUSW: KY 10,0 | hier: MW 10 |
| FEHL: M 12.0 | Fehlerbit = 1, wenn Uhr nicht korrekt angelaufen ist |
| : L KM 00000010 00110000 | Statuswort vorbesetzen |
| : T MW 10 | (hier z.B.:Betriebsstundenzähler freigeben, letzter RUN-STOP-Übergang wird abgespeichert, Uhrzeit wird im STOP der CPU aktualisiert) |
| :BE | |

FB 101

| FB 101 AWL | Erläuterung |
|---|---|
| NAME:UHR-INIT | Uhr initialisieren |
| BEZ:TUDA E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KC | |
| BEZ:NUDA E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KY | |
| BEZ:TUSW E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KC | |
| BEZ:NUSW E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KY | |
| BEZ:FEHL E/A/D/B/T/Z: A BI/BY/E/D: BI | |
| :LW =TUDA | Typ des Operandenbereichs für |
| :T MW 250 | Uhrdaten |
| :LW =NUDA | Anfangsadresse Uhrdatenbereich |
| :T MW 251 | |
| :LW =TUSW | Typ des Operandenbereichs für |
| :T MW 253 | Statuswort |
| :LW =NUSW | Adresse des Statuswortes |
| :T MW 254 | |
| : | |
| :L KHEEFF | Endadresse Quellbereich MB255) |
| :L KHEA15 | Endadresse Zielbereich (BS10) |
| :TNB 6 | MB250-255 in BS8-10 transfer. |
| :L KF+0 | Schmiermerker löschen |
| :T MW 250 | |
| :T MW 252 | |
| :T MW 254 | |
| :L BS 11 | Ist Uhr richtig angelaufen? |
| :L KH0003 | |
| :!=F | |
| :RB =FEHL | Fehlerbit rücksetzen |
| :BEB | |
| :S =FEHL | Fehlerbit setzen |
| :BE | |

DB 2

| DB 2 AWL | Erläuterung |
|---|---|
| 0:KH=0003 | --,Wochentag/ Akt.Uhrzeit |
| 1:KH=1402 | Tag, Monat |
| 2:KH=8908 | Jahr, Stunde + AM/PM-Bit |
| 3:KH=0000 | Minute, Sekunde |
| 4:KH=0103 | Schaltjahr, Wochentag/ Stellw. Uhr |
| 5:KH=1402 | Tag, Monat |
| 6:KH=8908 | Jahr, Stunde + AM/PM-Bit |
| 7:KH=0000 | Minute, Sekunde |
| 8:KH=0003 | --,Wochentag/ Weckzeit (Stellw.) |
| 9:KH=1402 | Tag, Monat |
| 10:KH=0009 | --,Stunde+AM/PM-Bit |
| 11:KH=0000 | Minute, Sekunde |
| 12:KH=0000 | --,Sekunden/Akt. Betriebsstd. |
| 13:KH=0001 | Minuten, Stunden |
| 14:KH=0000 | Stunden x 100, Stunden x 10 000 |
| 15:KH=0000 | --,Sekunden/Stellw.Betriebsstd. |
| 16:KH=0012 | Minuten, Stunden |
| 17:KH=0000 | Stunden x 100,Stunden x 10 000 |
| 18:KH=0000 | --Wochentag//Uhr nach STP/RUN |
| 19:KH=0000 | Tag, Monat |
| 20:KH=0000 | Jahr, Stunde |
| 21:KH=0000 | Minute, Sekunde |
| 22: | |

**BS 12 (Korrekturwert)**

Zur Kompensation der Ungenauigkeit infolge des Temperatureinflusses können Sie einen Korrekturwert in das Systemdatenwort BS 12 eintragen.

Der Korrekturwert (in Sekunden) bezieht sich auf die Laufzeit von 30 Tagen; d.h. wenn Sie feststellen, dass die Uhr in 30 Tagen z.B. um 20 Sekunden nachgeht, ist der Korrekturwert +20:

Intern kompensiert das Betriebssystem die Uhr stündlich mit einem Wert, der kleiner ist als eine Sekunde. Dies gewährleistet, dass die Uhr keine Sekunde überspringt.

Die Kompensation arbeitet, unabhängig von der eingestellten Betriebsart, sowohl im STOP- als auch im RUN-Zustand.

Den Korrekturwert müssen Sie im Format "KF" angeben.

Wertebereich: -400...0...+400 (bei "0" keine Korrektur).

Nach URLÖSCHEN ist der Wert "0" in BS 12 voreingestellt.

Bei ungültigem Korrekturwert setzt das Betriebssystem Bit Nr.15 in BS 11. In diesem Fall ist der Korrekturwert "0".

**Bei NETZ AUS wird die Uhrzeit nicht korrigiert.**

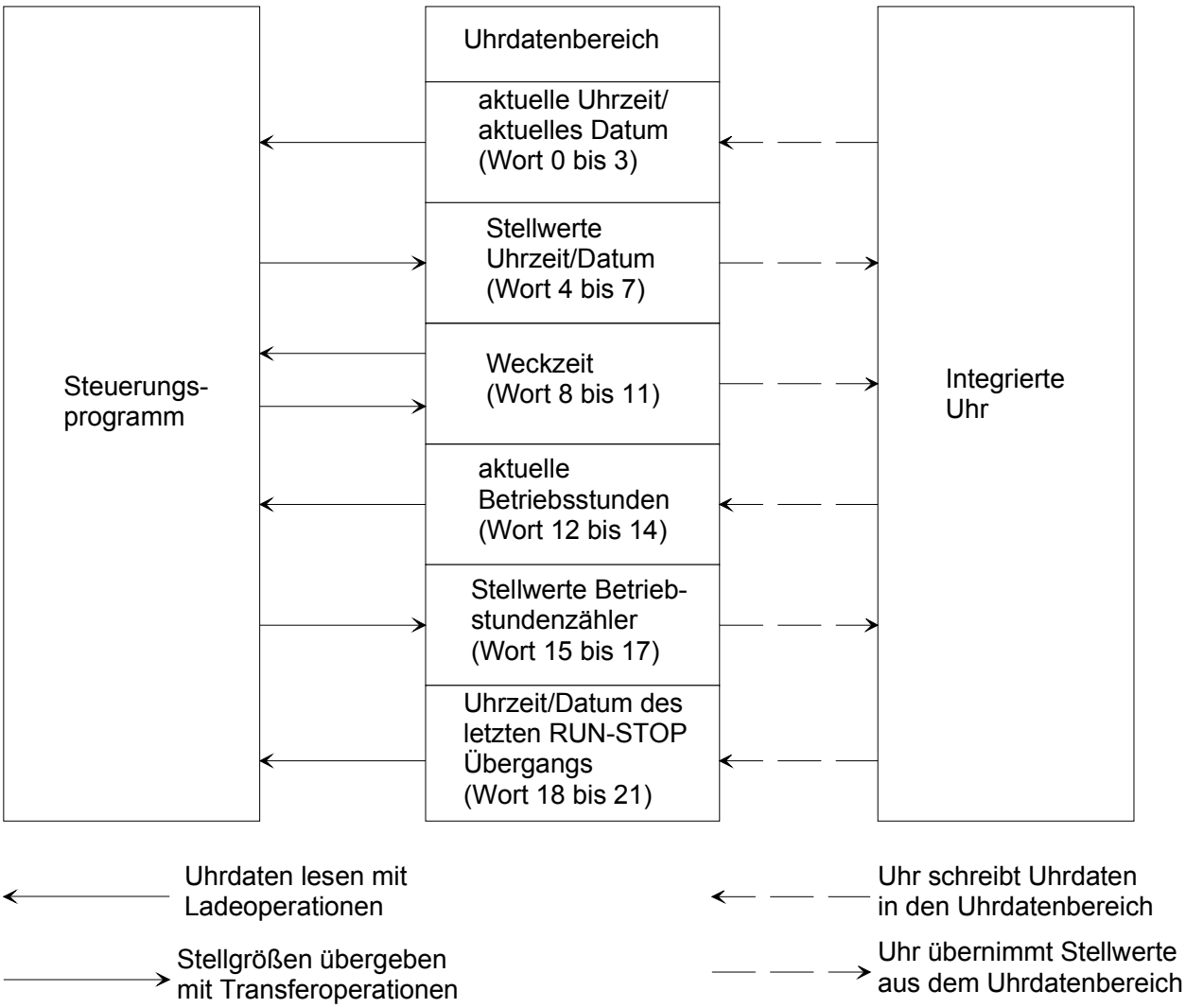
# Aufbau des Uhrdatenbereichs

Die Lage des Uhrdatenbereichs ist in den Systemdaten BS 8 und BS 9 zu hinterlegen. Der Datenaustausch zwischen Steuerungsprogramm und integrierter Uhr erfolgt über den Uhrdatenbereich.

Im Uhrdatenbereich (Merkerbereich oder Datenbaustein) hinterlegt die integrierte Uhr die aktuelle Uhrzeit, das Datum und den aktuellen Stand des Betriebsstundenzählers.

Stellwertvorgaben für Weckfunktion und Betriebsstundenzähler werden vom Steuerungsprogramm im selben Uhrdatenbereich hinterlegt.

Das Steuerungsprogramm kann nur den Uhrdatenbereich lesen oder beschreiben, nie direkt auf die Uhr zugreifen. Die nachfolgende Abbildung soll dies nochmals verdeutlichen.




Zum Stellen der Uhr sind nur die Daten zu übergeben, die für die jeweilige Funktion erforderlich sind. Wenn Sie z.B. nur Daten für die Uhrfunktion ändern wollen, müssen Sie die Daten für die Weckfunktion oder für den Betriebsstundenzähler nicht angeben.

**Uhrdatenbereich** Nachfolgende Tabelle gibt Auskunft über den Aufbau des Uhrdatenbereichs:

| Wort-Nr. | Bedeutung | Wort links | Wort rechts |
|---|---|---|---|
| 0 | Aktuelle Uhrzeit/ | --- | Wochentag |
| 1 | Aktuelles Datum | Tag | Monat |
| 2 | | Jahr | Bit 7: (AM/PM), 6..0: Stunde |
| 3 | | Minute | Sekunde |
| 4 | Stellwertvorgabe | | Wochentag |
| 5 | Uhrzeit/Datum | Tag | Monat |
| 6 | | Jahr | AM/PM(Bit Nr.7)<br>Stunde |
| 7 | | Minute | Sekunde |
| 8 | Weckzeit | --- | Wochentag |
| 9 | | Tag | Monat |
| 10 | | --- | Bit 7: (AM/PM), 6..0: Stunde |
| 11 | | Minute | Sekunde |
| 12 | Aktuelle Betriebsstunden | - | Sekunden |
| 13 | | Minuten | Stunden |
| 14 | | Stunden x 100 | Stunden x 10 000 |
| 15 | Stellwertvorgabe | - | Sekunden |
| 16 | Betriebsstundenzähler | Minuten | Stunden |
| 17 | | Stunden x 100 | Stunden x 10 000 |
| 18 | Uhrzeit/Datum nach | --- | Wochentag |
| 19 | einem RUN-STOP- | Tag | Monat |
| 20 | Übergang bzw. NETZ-Aus (nur | Jahr | Bit 7: (AM/PM), 6..0: Stunde |
| 21 | wenn im Statuswort Bit Nr. 5 = 1) | Minute | Sekunde |

**Verschieben des Uhrdatenbereichs**

Wird der Uhrdatenbereich an das Ende der einzelnen Bereiche (Merker, Datenbaustein) gelegt und ist nicht mehr genügend freier Speicherplatz für den Uhrdatenbereich vorhanden, wird der Fehler durch setzen des Bit 14 im BS 11 signalisiert.

Liegen die Uhrdaten im nichtremanenten Merkerbereich, gehen nach NETZ-AUS und NEUSTART alle Stellwerte und der Zeitpunkt des letzten RUN-STOP-Überganges verloren!

Berücksichtigen Sie, dass die Lage des Uhrdatenbereichs von Ihnen festgelegt werden kann und die angegebenen Wortnummern relative Angaben sind.

Liegt Ihr Uhrdatenbereich in einem Datenbaustein und beginnt nicht mit dem DW 0 sondern mit DW X, müssen Sie zur Wortnummer in der Tabelle den Wert X addieren.

*Beispiel*:

Ihr Uhrdatenbereich beginnt bei DW 124. Die Daten für Uhrzeit/Datum werden dann von DW 124 bis 127 abgelegt.

**Besonderheiten**

Folgende Besonderheiten sind zu berücksichtigen:

- Im Uhrdatenbereich sind die Einträge dezimal, also BCD-codiert einzutragen. Das Datenformat "KZ" lädt eine Konstante BCD-codiert in den AKKU 1 und eignet sich so besonders für die Stellwertvorgabe.
- Schaltjahre werden von der Uhr selbständig berücksichtigt.
- Sie können durch Beeinflussung von Bit Nr. 1 im Statuswort wählen, ob die Uhr im 12-Stunden- oder im 24-Stunden-Modus laufen soll. Das AM/PM-Flag (0 = AM; 1 = PM) spielt nur im 12-Stunden-Modus der Hardwareuhr eine Rolle. Es entspricht Bit 7 folgender Wörter. Wort 2, Wort 6, Wort 10, Wort 20.
  In dieser Betriebsart können die Stunden und das AM/PM - Flag bei der Stellwertvorgabe der Uhr und der Weckzeit nicht unabhängig voneinander gesetzt werden.
  Im 24-Stunden-Modus wird ein gesetztes AM/PM-Flag bei der Stellwertvorgabe der Uhr und Weckzeit berücksichtigt und verursacht ein Setzen des jeweiligen Fehlerbits.

**Definitionsbereich für Stellwerte**

Die Stellwertvorgaben müssen innerhalb der in folgender Tabelle angegebenen Definitionsbereiche liegen:

| Größe | erlaubte Parameter | Größe | erlaubte Parameter |
|---|---|---|---|
| Sekunden | 0...59 | Tag | 1...31 |
| Minuten | 0...59 | Monat | 1...12 |
| Stunden | im 24h Modus:<br>0...23<br>bei AM 1...12<br>(12=0 Uhr)<br>bei PM 81...92<br>(92=12 Uhr mittags)<br>mit gesetztem<br>AM/PM-Bit)<br>0...99 bei Vorgabe<br>Betriebstunden-<br>zähler | Jahr | 0...99 |
| Wochentag | 0...7<br>1 = Sonntag<br>2 = Montag<br>3 = Dienstag<br>4 = Mittwoch<br>5 = Donnerstag<br>6 = Freitag<br>7 = Samstag | | |

# Aufbau des Statusworts

**Funktion**

Das Statuswort kann einerseits abgefragt werden, um Fehler z.B. bei der Stellwertvorgabe zu erkennen, andererseits können durch Veränderung bestimmter Bits des Statusworts gezielt Übernahme- oder Leseoperationen gesperrt bzw. freigegeben werden.

Außerdem ist das Verhalten der Uhr beim Übergang der CPU vom RUN- in den STOP-Zustand bzw. bei NETZ-Aus mit den dafür vorgesehenen Bits (Flags) bestimmbar.

Das Statuswort kann im Merkerbereich oder in einem Datenbaustein liegen. Die Lage ist in Systemdatum BS 9 und BS 10 festzulegen.

Die Bits des Statuswortes werden unterschieden in:

- Uhr-Flags
- Betriebssystem-Flags
- Betriebsstundenzähler-Flags,
- Weckzeit-Flags.

**Bits setzen/ rücksetzen**

Die Bits "Stellwerte übernehmen" (Bit Nr. 2, 10, 14 des Statuswortes) werden vom Betriebssystem zurückgesetzt, wenn:

- die Stellwerte übernommen wurden,
- die Stellwerte nicht übernommen wurden, weil sie außerhalb des zulässigen Bereichs lagen. In diesem Fall wird zusätzlich das entsprechende Fehlerbit (Bit Nr. 0, 8, 12 des Statuswortes) gesetzt.

Die Bits "Stellwerte übernehmen" (Bit Nr. 2, 10, 14 des Statuswortes) werden vom Betriebsystem nicht zurückgesetzt, wenn:

- die Systemdaten für die Uhr falsch oder nicht vorhanden sind
- der Uhrdatenbereich zu klein ist
- die Uhr defekt ist (Hardwarefehler).

**Abfrage des Statusworts**

In einem Datenbaustein können Sie die einzelnen Bits eines Datenworts mit der Operation "P<Datenwortnummer> <Bitnummer>" abfragen. Im Merkerbereich fragen Sie die einzelnen Bits ab, indem Sie die <Byte-Adresse>und die <Bitnummer> angeben.

*Beispiel:*

Das Statuswort ist im DW 13 abgelegt. Sie wollen überprüfen, ob die eingestellte Weckzeit erreicht ist.

Die Abfrage erfolgt durch die Anweisung "P D 13.13"

Wenn das Statuswort im MW 13 abgelegt ist, lautet die gleiche Abfrage "U M 13.5"

**Aufbau Statuswort**

| Bit-Nr. | Funktion |
|---|---|
| | **Uhr-Flags** |
| 0 | 0: kein Fehler in der Stellwertvorgabe<br>1: Fehler in der Stellwertvorgabe |
| 1 | 0: 24h - Darstellung (Uhr-Modus)<br>1: 12h - Darstellung (Uhr-Modus) |
| 2 | 0: Stellwerte nicht übernehmen<br>1: Stellwerte übernehmen |
| 3 | 0: Uhrzeit kann nicht gelesen werden<br>1: Uhrzeit kann gelesen werden |
| | **Betriebssystem-Flags** |
| 4 | Betriebsart STOP<br>0: Die Uhr aktualisiert den Uhrdatenbereich nicht.<br>Wort 0 bis 3 enthalten den Zeitpunkt des letzten RUN-Stop-Übergangs.<br>1: Die Uhr aktualisiert im Uhrdatenbereich nur die Worte 0 bis 3 (aktuelle Uhrzeit/ aktuelles Datum). Die Uhr kann mit der PG-Funktion "STEUERN VAR" gestellt werden.<br>Betriebsart RUN<br>0/1: Die Uhr aktualisiert fortlaufend den Uhrdatenbereich (Wort 0 bis 17). |
| 5 | Betriebsart STOP<br>0: Wort 18 bis 21 werden nicht benutzt.<br>1: Wort 18 bis 21 enthalten den Zeitpunkt des letzten RUN-STOP-Übergangs bzw. den Zeitpunkt des letzten NETZ-AUS, wenn zusätzlich Bit 4 gesetzt ist.<br>Betriebsart RUN<br>0: Wort 18 bis 21 werden nicht benutzt.<br>1: Wort 18 bis 21 enthalten den Zeitpunkt des letzten RUN-STOP-Übergangs bzw. den Zeitpunkt des letzten NETZ-AUS. |
| 6 ... 7 | reserviert |
| | **Betriebsstundenzähler-Flags** |
| 8 | 0: kein Fehler in der Stellwertvorgabe<br>1: Fehler in der Stellwertvorgabe |
| 9 | 0: Betriebsstundenzähler sperren zum Auslesen<br>1: Betriebsstundenzähler freigeben |
| 10 | 0: Stellwerte nicht übernehmen<br>1: Stellwerte übernehmen |
| 11 | reserviert |
| | **Weckuhr-Flags** |
| 12 | 0: kein Fehler in der Stellwertvorgabe<br>1: Fehler in der Stellwertvorgabe |
| 13 | 0: eingestellte Weckzeit nicht erreicht<br>1: eingestellte Weckzeit erreicht |
| 14 | 0: Stellwerte nicht übernehmen<br>1: Stellwerte übernehmen |
| 15 | reserviert |

# Uhr stellen und lesen

**Stellwertvorgaben übernehmen**

Nachdem Sie Ihre Stellwerte in den Uhrdatenbereich transferiert haben, veranlassen Sie durch Setzen von Bit 2 des Statusworts die Übernahme der Stellwertvorgaben.

**Stellwertvorgabe nicht übernehmen**

Soll eine Stellwertvorgabe (Weckzeit oder Betriebsstundenzähler) beim Stellen nicht in die Uhr übernommen werden bzw. der aktuelle Wert nicht geändert werden, ist für diese Größe der Zahlenwert "FFh" einzugeben.

**Reaktion bei Fehleingabe**

Sobald ein Stellwertvorgabe über den Definitions-Bereich hinaus geht, führt dies zu einer Fehlermeldung, die in einem Statuswort angezeigt wird. Die Uhr läuft mit den alten Werten weiter. Die Fehlermeldung wird zurückgesetzt. sobald der Fehler behoben ist.

**Uhrdaten in Merkerbereich**

Liegt Ihr Uhrdatenbereich im Merkerbereich, so müssen Sie zur Ermittlung der entsprechenden Wortadresse die in der Uhrdatenbereich-Tabelle angegebene Wort-Nr. mit dem Faktor 2 multiplizieren.

Beginnt Ihr Uhrdatenbereich nicht bei MW 0, müssen Sie zusätzlich noch diesen Wert addieren.

*Beispiel***:**

Sie legen den Uhrdatenbereich im Operandenbereich Merker ab MW 0 ab. Die Daten für den Betriebsstundenzähler werden somit beginnend mit der Adresse MW 24 abgelegt.

**Aktuelle Uhrzeit / aktuelles Datum lesen**

Die aktuellen Daten sind im Uhrdatenbereich in den ersten vier Datenworten abgelegt. Von dort können Sie mit Ladeoperationen ausgelesen werden.

Um eine korrekte Uhrzeit lesen zu können, muss vor dem Lesezugriff im Steuerungsprogramm das Bit 3 des Statuswortes gesetzt werden.

Der Uhrdatenbereich wird bei gesetztem Bit 3 nicht mehr aktualisiert. Nach dem Lesen der Uhr müssen Sie das Bit wieder zurücksetzen.

**Bit 3 setzen ⇒ Uhrzeit lesen ⇒ Bit 3 rücksetzen!**

**Ablage von Uhrzeit/Datum nach RUN-STOP**

Sie haben die Möglichkeit, einen RUN-STOP Übergang bzw. NETZ-AUS zeitlich festzuhalten, auch wenn das AG sich inzwischen wieder in der Betriebsart RUN befindet.

Die Uhrzeit und das Datum des letzten RUN-STOP Übergangs bzw. NETZ-AUS stehen in den Worten 18 bis 21.

*Auszug aus Uhrdatenbereich*

| Wort-Nr. | Bedeutung | Wort links | Wort rechts |
|---|---|---|---|
| 18 | Uhrzeit/Datum nach | --- | Wochentag |
| 19 | einem RUN-STOP- | Tag | Monat |
| 20 | Übergang bzw. NETZ-Aus (nur | Jahr | Bit 7: (AM/PM), 6..0: Stunde |
| 21 | wenn im Statuswort Bit Nr. 5 = 1) | Minute | Sekunde |

Haben mehrere RUN-STOP Übergänge stattgefunden, bevor Sie diesen Uhrdatenbereich ausgelesen haben, können Sie nur den Zeitpunkt des letzten Übergangs feststellen.

**Hinweis!**

Diese Uhrdatenbereich wird nur beschrieben, wenn:

- Bit 5 im Statuswort auf 1 gesetzt ist,
- ein RUN-STOP Übergang bzw. NETZ-AUS stattgefunden hat
- im Operandenbereich der Speicherplatz zur Verfügung steht.

Haben Sie nicht genügend Speicherplatz für diesen Uhrdatenbereich zur Verfügung, können Sie diesem Bereich nicht oder nur teilweise nutzen. Dies hat keinerlei sonstige Auswirkungen.

**Beispiel: Uhrzeit und Datum einstellen**

Die Uhr soll mit folgenden Daten gestellt werden: Di 01.03.88; 12.00.00. Das Statuswort belegt das Merkerwort 10 und die Uhrdaten werden im DB 2 ab Datenwort 0 abgelegt. Die Stellwerte für die Uhrdaten werden übergeben:

- Mit der PG-Funktion "STEUERN VAR", wenn das AG in "RUN",
- Mit der PG-Funktion "STEUERN VAR", wenn das AG in "STOP" und Statuswort Bit 4 =1.

**Hinweis!**

Bei "STEUERN VAR" müssen Sie zuerst die Uhrdaten und zuletzt das Statuswort angeben.

*Uhrdatenbereich liegt in DB2*

| Operand | Signalzustand | Bedeutung |
|---|---|---|
| DB2 | | |
| DW 4 | KH=0003 | Wochentag (Dienstag) |
| DW 5 | KH=0103 | Tagesdatum (01) und Monat (03) |
| DW 6 | KH=8812 | Jahreszahl (88) und Stunde (12) |
| DW 7 | KH=0000 | Minute (00) und Sekunde (00) |
| MW 10 | KH=0014 | In "STOP" und "RUN":<br>Bit 4=1: Uhrdatenbereich wird in "STOP" aktualisiert |
| | oder | Bit 2=1: Stellwerte übernehmen<br>nur in "RUN": |
| MW 10 | KH=0004 | Bit 4=0: Uhrdaten werden nicht in "STOP" aktualisiert.<br>Bit 2=1: Stellwerte übernehmen |

Abhängig vom Eingang 12.1 werden Stellwerte für Uhrzeit und Datum übernommen. Diese Stellwerte müssen Sie vor dem Setzen von Eingang 12.1 in die Merkerbytes 120 bis 127 transferieren. Werte die nicht verändert werden sollen, sind mit FFh vorzubelegen. Mit Eingang 14.0 kann der Modus der Uhr bestimmt werden (1=12-Stunden-Modus). Eingang 13.0 ist das AM/PM-Bit, das im 12-Stunden-Modus der Uhr berücksichtigt wird.

Der Uhrdatenbereich liegt im DB 2 ab DW 0, das Statuswort ist MW 10.

OB 1

| OB 1 AWL | | Erläuterung |
|---|---|---|
| | | **Uhrzeit und Datum stellen** |
| | | Werte für Uhrzeit und Datum zuerst in MB 121 bis MB 127 transferieren |
| | :U E 12.1 | Anstoß des Uhrstellens mit Setzen von M 20.0 |
| | :S M 20.0 | (wird im FB10 zurückgesetzt) |
| | : | |
| | :SPA FB 10 | |
| NAME | :UHR-STEL | |
| WOTG | :MB 121 | Wochentag |
| TAG | :MB 122 | Tag |
| MON | :MB 123 | Monat |
| JAHR | :MB 124 | Jahr |
| STD | :MB 125 | Stunde |
| AMPM | :E 13.0 | AM/PM-Bit (12-h-Modus) |
| MIN | :MB 126 | Minuten |
| SEK | :MB 127 | Sekunden |
| FEHL | :M 12.1 | Fehlerbit |
| MODE | :E 14.0 | 12h-Mode: E 14.0 =1 |
| | :BE | |

FB 10

| FB 10 AWL | | Erläuterung |
|---|---|---|
| NAME | :UHR-STEL | **Uhr stellen** |
| BEZ | :WOTG E/A/D/B/T/Z: E | BI/BY/W/D: BY |
| BEZ | :TAG E/A/D/B/T/Z: E | BI/BY/W/D: BY |
| BEZ | :MON E/A/D/B/T/Z: E | BI/BY/W/D: BY |
| BEZ | :JAHR E/A/D/B/T/Z: E | BI/BY/W/D: BY |
| BEZ | :STD E/A/D/B/T/Z: E | BI/BY/W/D: BY |
| BEZ | :AMPM E/A/D/B/T/Z: E | BI/BY/W/D: BI |
| BEZ | :MIN E/A/D/B/T/Z: E | BI/BY/W/D: BY |
| BEZ | :SEK E/A/D/B/T/Z: E | BI/BY/W/D: BY |
| BEZ | :FEHL E/A/D/B/T/Z: A | BI/BY/W/D: BI |
| BEZ | :MODE E/A/D/B/T/Z: E | BI/BY/W/D: BI |
| | :U =MODE | 24h-Modus =0; 12h_MOD.=1 |
| | := M 11.1 | Uhrmodus Statuswort |
| | :UN M 20.0 | Merker ist rückgesetzt, |
| | SPB =M001 | wenn Stellwerte bereits in den Uhrdatenbereich eingelesen wurden |
| | :R M 20.0 | |
| | :A DB 2 | Uhrdatenbereich |
| | :L =WOTG | Wert für Wochentag abgel. |
| | :T DR 4 | |
| | :L =TAG | Wert für Tag ablegen |
| | :T DL 5 | |
| | :L =MON | Wert für Monat ablegen |
| | :T DR 5 | |
| | :L =JAHR | Wert für Jahr ablegen |
| | :T DL 6 | |
| | :L =STD | Wert für Stunde ablegen |
| | :ON =AMPM | wenn 12h-Modus eingest. |
| | :ON =MODE | und AMPM-Bit = 1 |
| | :SPB =VORM | dann wird entspr. Bit |
| | :L KH 0080 | im Uhrdatenbereich |
| | :OW | gesetzt |
| VORM | :T DR 6 | |
| | :L =MIN | Wert für Minute ablegen |
| | :T DL 7 | |
| | :L =SEK | Wert für Sekunde ablegen |
| | :T DR 7 | |
| | :UN M 11.2 | Stellwerte übernehmen |
| | :S M 11.2 | (Statuswort ist MW 10) |
| | :L KT 020.1 | Überwachungszeit starten |
| | :SV T 10 | |
| M001 | :U T 10 | BEB, wenn Überwachungsz. |
| | :BEB | noch nicht abgelaufen |
| | :UN M 11.2 | wurden Stellwerte übernommen? |
| | :SPB =M002 | wenn ja |
| | :S =FEHL | Fehlerbit setzen |
| | :BEA | |
| M002 | :UN M 11.0 | Fehler bei Stellwertvorgabe |
| | :RB =FEHL | Fehlerbit rücksetzen |
| | :BEB | |
| | :S =FEHL | Fehlerbit setzen |
| | :BE | |

**Beispiel: Uhrzeit und Datum lesen**

Abhängig von einem externen Ereignis, hier simuliert durch eine positive Flanke am Eingang 12.0, wird die Uhrzeit in den Merkerbytes 30 bis 36 abgespeichert. Im Merker 13.1 wird angezeigt, in welchem Modus die Uhr arbeitet. Merker 13.0 ist im 12-Stunden-Modus das AM/PM-Bit. Der Uhrdatenbereich liegt im DB 2 ab DW 0,das Statuswort ist MW 10.

OB1

| OB 1 AWL | | Erläuterung |
|---|---|---|
| | : | **Uhrzeit und Datum lesen** |
| | : | |
| | : | |
| | :U E 12.0 | Bei Positive Flanke von E 12.0 |
| | :UN M 0.1 | soll Uhrzeit und Datum |
| | := M 0.0 | in MB 30 - MB 36 |
| | :U E 12.0 | abgespeichert werden |
| | := M 0.1 | |
| | : | |
| | : | |
| | :U M 0.0 | Flankenmerker |
| | :SPB FB 13 | |
| NAME | :UHR-LES | |
| WOTG | :MB 30 | Wochentag |
| TAG | :MB 31 | Tag |
| MON | :MB 32 | Monat |
| JAHR | :MB 33 | Jahr |
| STD | :MB 34 | Stunde |
| AMPM | :M 13.0 | nachmittags im 12h-Mod |
| MIN | :MB 35 | Minuten |
| SEK | :MB 36 | Sekunden |
| MODE | :M 13.1 | im 12h-Modus |
| | :BE | |

FB 13

| FB 13 AWL | | Erläuterung |
|---|---|---|
| NAME | :UHR-STEL | **Uhr stellen** |
| BEZ | :WOTG E/A/D/B/T/Z: A | BI/BY/W/D: BY |
| BEZ | :TAG E/A/D/B/T/Z: A | BI/BY/W/D: BY |
| BEZ | :MON E/A/D/B/T/Z: A | BI/BY/W/D: BY |
| BEZ | :JAHR E/A/D/B/T/Z: A | BI/BY/W/D: BY |
| BEZ | :STD E/A/D/B/T/Z: A | BI/BY/W/D: BY |
| BEZ | :AMPM E/A/D/B/T/Z: A | BI/BY/W/D: BI |
| BEZ | :MIN E/A/D/B/T/Z: A | BI/BY/W/D: BY |
| BEZ | :SEK E/A/D/B/T/Z: A | BI/BY/W/D: BY |
| BEZ | :MODE E/A/D/B/T/Z: A | BI/BY/W/D: BI |
| | | |
| | :UN M 11.3 | Uhrzeit kann gelesen w. |
| | :S M 11.3 | (Bit 3 im Statuswort setzen |
| | :A DB 2 | |
| | :L DR 0 | Wochentag |
| | :T =WOTG | |
| | :L DL 1 | Tag |
| | :T =TAG | |
| | :L DR 1 | Monat |
| | :T =MON | |
| | :L DL 2 | Jahr |
| | :T =JAHR | |
| | :L DR 2 | Stunde |
| | :L KH 007F | AMPM-Bit ausblenden |
| | :UW | nur im 12h-Modus relevant |
| | :T =STD | |
| | P D 2.7 | AMPM-Bit anzeigen |
| | := =AMPM | nur im 12h-Modus relevant |
| | :L DL 3 | Minute |
| | :T =MIN | |
| | :L DR 3 | Sekunde |
| | :T =SEK | |
| | :U M 11.3 | Uhrzeit wird wieder |
| | :R M 11.3 | aktualisiert |
| | :U M 11.1 | Uhr-Modus anzeigen |
| | := =MODE | Mode=1, bei 12h-Modus |
| | :BE | |

# Weckfunktionen

**Übersicht**

Über die Weckfunktionen haben Sie die Möglichkeit im Uhrdatenbereich eine Weckzeit einzustellen, die bei Erreichen das Bit 13 im Statuswort setzt und über Ihr Anwenderprogramm ausgewertet werden kann.

Wie unter "Uhr stellen und lesen" sind die Weckdaten BCD-codiert im Uhrdatenbereich einzutragen.

*Auszug aus Uhrdatenbereich*

| Wort-Nr. | Bedeutung | Wort links | Wort rechts |
|---|---|---|---|
| 8 | Weckzeit | --- | Wochentag |
| 9 | | Tag | Monat |
| 10 | | --- | Bit 7: (AM/PM), 6..0: Stunde |
| 11 | | Minute | Sekunde |

Die Übernahme der Stellwerte der Weckfunktion in die Uhr wird durch das Bit 14 im Statuswort veranlasst.

Fehlerhafte Stellwerte werden mit Bit 12 im Statuswort angezeigt.

Die Weckzeit kann jederzeit gelesen werden

**periodische Weckfunktion**

Tragen Sie in der Weckzeit in einem Byte den Zahlenwert "255" (dezimal) oder "FFh" ein, wird dieses Byte bei der Beurteilung "Weckzeit erreicht" nicht berücksichtigt.

Dies ermöglicht zum Beispiel eine bequeme Programmierung eines sich täglich wiederholenden Alarms, indem man in den Stellgrößen "Wochentag", "Tag" und "Monat" den Wert "255" (dezimal) oder "FFh" einträgt.

**Ablauf der Weckzeit**

Nach Ablauf der Weckzeit wird das Bit 13 im Statuswort gesetzt.

Das Bit 13 bleibt solange gesetzt, bis Sie es im Steuerungsprogramm zurücksetzen.

**Vorsicht!**

Wird die Weckzeit in der Betriebsart STOP oder im Zustand NETZ-AUS erreicht, kann das Weckzeit-Bit nicht ausgewertet werden. Es wird im ANLAUF immer gelöscht!

**Beispiel**

**Weckzeit stellen und auswerten.**

Abhängig vom Zustand des Eingangs 12.2 werden im Beispielprogramm die Stellwerte für die Weckzeit übernommen. Die Stellwerte sind vor dem Setzen des Eingangs 12.2 von Ihnen in die Merkerbytes 130 bis 135 zu transferieren.

Werte, die nicht berücksichtigt werden sollen, sind mit FFh vorzubelegen.

Mit Eingang 14.0 wird der Modus der Uhr eingestellt. Mit Eingang 13.0 geben Sie das AM/PM-Bit für den 12-Stunden-Modus vor.

Ist die vorgegebene Weckzeit erreicht, wird Merker 13.2 gesetzt.

Fehler bei der Weckzeitvorgabe werden im Merker 12.2 angezeigt.

Die Uhrdaten sind im DB 2 ab DW 0 abgelegt, das Statuswort ist MW 10.

OB1

| OB 1 AWL | | Erläuterung |
|---|---|---|
| | : | **Weckzeit stellen und auslesen** |
| | : | |
| | : | Werte zuerst in MB 130 bis MB 135 transferieren |
| | : | |
| | :U E 12.2 | Anstoß des Weckzeit-Stellens |
| | :S M 20.1 | mit setzen von M20.1 (wird in FB 11 zurückgesetzt) |
| | : | |
| | : | |
| | :SPA FB 11 | |
| NAME | :WECKZ-ST | |
| WOTG | :MB 130 | Wochentag |
| TAG | :MB 131 | Tag |
| MON | :MB 132 | Monat |
| STD | :MB 133 | Stunde |
| AMPM | :E 13.0 | AMPM-Bit |
| MIN | :MB 134 | Minuten |
| SEK | :MB 135 | Sekunden |
| FEHL | :M 12.2 | Fehlerbit |
| ALRM | :M 13.2 | Anzeige Weckzeit err. |
| MODE | :E 14.0 | im 12h-Modus |
| | :BE | |

FB 11

| FB 11 AWL | | Erläuterung |
|---|---|---|
| NAME | :WECKZ-ST | **Weckzeit stellen** |
| BEZ | :WOTG E/A/D/B/T/Z: | E BI/BY/W/D: BY |
| BEZ | :TAG E/A/D/B/T/Z: | E BI/BY/W/D: BY |
| BEZ | :MON E/A/D/B/T/Z: | E BI/BY/W/D: BY |
| BEZ | :STD E/A/D/B/T/Z: | E BI/BY/W/D: BY |
| BEZ | :AMPM E/A/D/B/T/Z: | E BI/BY/W/D: BI |
| BEZ | :MIN E/A/D/B/T/Z: | E BI/BY/W/D: BY |
| BEZ | :SEK E/A/D/B/T/Z: | E BI/BY/W/D: BY |
| BEZ | :FEHL E/A/D/B/T/Z: | A BI/BY/W/D: BI |
| BEZ | :ALRM E/A/D/B/T/Z: | A BI/BY/W/D: BI |
| BEZ | :MODE E/A/D/B/T/Z: | E BI/BY/W/D: BI |
| | | |
| | :U =MODE | 12h-Modus = 1 |
| | := M 11.1 | (Uhrmodus einstellen) |
| | :U M 10.5 | Weckzeit erreicht anz. |
| | :S =ALRM | (Bit 13 im Statuswort) |
| | :R M 10.5 | Bit nach Auswertung rücksetzen |
| | :UN M 20.1 | Merker ist rückgesetzt, |
| | :SPB =M001 | wenn Stellwerte bereits |
| | :R M 20.1 | in den Uhrdatenbereich |
| | : | eingelesen wurden |
| | : | |
| | :A DB 2 | Uhrdatenbereich |
| | :L =WOTG | Wert für Wochentag |
| | :T DR 8 | |
| | :L =TAG | Wert für Tag |
| | :T DL 9 | |
| | :L =MON | Wert für Monat |
| | :T DR 9 | |
| | :L =STD | Wert für Stunde |
| | :ON =AMPM | wenn AMPM=1 (nachmittags |
| | :ON =MODE | 12h-Modus eingestellt, |
| | :SPB =VORM | dann wird entspr. Bit |
| | | im Uhrdatenbereich gesetzt |
| | :L KH 0080 | |
| | :OW | |
| VORM | :T DR 10 | |
| | :L =MIN | Wert für Minute |
| | :T DL 11 | |
| | :L =SEK | Wert für Sekunde |
| | :T DR 11 | |
| | :UN M 10.6 | Stellwerte übernehmen |
| | :S M 10.6 | (Bit 14 im Statuswort MW10) |
| | :L KT 020.1 | Überwachungszeit starten |
| | :SV T 11 | |
| M001 | :U T 11 | BEB, wenn Zeit noch nicht |
| | | abgelaufen |
| | :BEB | |
| | :UN M 10.6 | wurden Stellwerte übernommen? |
| | :SPB =M002 | Sprung nach M002 |
| | :S =FEHL | Fehlerbit setzen |
| | :BEA | |
| M002 | :UN M 10.4 | Fehler? |
| | :RB =FEHL | Fehlerbit rücksetzen |
| | :BEB | BEB, wenn kein Fehler |
| | :S =FEHL | Fehler-Bit setzen |
| | :BE | |

# Betriebsstundenzähler

**Übersicht**

Im Betriebsstundenzähler wird die Zeitdauer erfasst, in der Ihre CPU im RUN-Zustand sich befindet. So können Sie zum Beispiel die Einschaltdauer eines Motors feststellen.

Durch Setzen von Bit 9 des Statusworts aktivieren Sie den Betriebsstundenzähler. Wie bei den Uhr- und Weckfunktionen können Sie auch hier über den Uhrdatenbereich Stellwerte BCD-codiert vorgeben.

Die aktuellen Betriebsstunden werden auch im Uhrdatenbereich abgelegt:

*Auszug aus Uhrdatenbereich*

| Wort-Nr. | Bedeutung | Wort links | Wort rechts |
|---|---|---|---|
| 12 | Aktuelle Betriebsstunden | - | Sekunden |
| 13 | | Minuten | Stunden |
| 14 | | Stunden x 100 | Stunden x 10 000 |
| 15 | Stellwertvorgabe | - | Sekunden |
| 16 | Betriebsstundenzähler | Minuten | Stunden |
| 17 | | Stunden x 100 | Stunden x 10 000 |

Soll bei der Stellwertvorgabe des Betriebstundenzählers eine Stellgröße nicht übernommen werden, kennzeichnen Sie das entsprechende Byte mit dem Zahlenwert "255" (dezimal) oder "FFh". Beim Stellen bleibt dann der im Betriebstundenzähler vorhandene Wert dieser Stellgröße erhalten.

**Betriebsstunden-zähler stellen**

Tragen Sie Ihre Stellwertvorgaben in den Uhrdatenbereich Wort-Nr. 15...17 ein und setzen Sie zur Übernahme Ihrer Daten in die Uhr Bit 10 des Statusworts.

Fehlerhafte Stellwerte werden mit Bit 8 im Statuswort angezeigt.

**Betriebsstunden abrufen**

Die aktuellen Daten sind im Uhrdatenbereich in den Worten 12 bis 14 abgelegt. Von dort können sie mit Ladeoperationen ausgelesen werden.

Um den Betriebsstundenzähler korrekt lesen zu können, muss vor dem Lesezugriff im Steuerungsprogramm das Bit 9 rückgesetzt werden. Der Uhrdatenbereich wird bei rückgesetztem Bit 9 nicht mehr aktualisiert. Nach dem Lesen der Uhr müssen Sie das Bit wieder setzen.

**Bit 9 rücksetzen $\Rightarrow$ Uhrzeit lesen $\Rightarrow$ Bit 9 setzen!**

**Beispiel: Stellen des Betriebsstunden-zählers**

Abhängig vom Zustand des Eingangs 12.3 sollen die Stellwerte für den Betriebsstundenzähler übernommen werden. Diese Werte müssen in die Merkerbytes 136 bis 140 transferieren, und zwar bevor der Eingang 12.3 gesetzt wird (im Beispielprogramm nicht durchgeführt).

Werte die nicht verändert werden sollen, sind mit FFh vorzubelegen.

Fehler bei der Stellwertvorgabe werden in Merker 12.3 angezeigt.

Der Uhrdatenbereich liegt im DB 2 ab DW 0, das Statuswort ist MW 10.

OB 1

| OB 1 AWL | | Erläuterung |
|---|---|---|
| | : | **Betriebstundenzähler stellen** |
| | : | |
| | : | Werte zuerst in MB 136 bis MB 140 transferieren |
| | : | |
| | :U E 12.3 | Anstoß der Übernahme der Stellwerte für Betriebstundenzähler |
| | :S M 20.2 | mit Setzen von M20.2 |
| | : | |
| | : | |
| | :SPA FB 12 | |
| NAME | :BETRST-S | |
| SEK | :MB 136 | Sekunden |
| MIN | :MB 137 | Minuten |
| STD0 | :MB 138 | Stunden |
| STD2 | :MB 139 | Stunden x 100 |
| STD4 | :MB 140 | Stunden x 10000 |
| FEHL | :M 12.3 | Fehlerbit |
| | :BE | |

FB 12

| FB 12 AWL | | Erläuterung |
|---|---|---|
| NAME | :BETRST-S | **Betriebstundenzähler stellen** |
| BEZ | :SEK E/A/D/B/T/Z: E | BI/BY/W/D: BY |
| BEZ | :MIN E/A/D/B/T/Z: E | BI/BY/W/D: BY |
| BEZ | :STD0 E/A/D/B/T/Z: E | BI/BY/W/D: BY |
| BEZ | :STD2 E/A/D/B/T/Z: E | BI/BY/W/D: BY |
| BEZ | :STD4 E/A/D/B/T/Z: E | BI/BY/W/D: BY |
| BEZ | :FEHL E/A/D/B/T/Z: A | BI/BY/W/D: BI |
| | | |
| | :UN M 20.2 | Merker ist rückgesetzt, |
| | :SPB =M001 | wenn Stellwerte bereits in den Uhrdatenbereich eingelesen wurden |
| | :R M 20.2 | |
| | : | |
| | :A DB 2 | Uhrdatenbereich |
| | :L =SEK | Wert für Sekunden |
| | :T DR 15 | |
| | :L =MIN | Wert für Minuten |
| | :T DL 16 | |
| | :L =STD0 | Wert für Stunden |
| | :T DR 16 | |
| | :L =STD2 | Wert für Stunden x 100 |
| | :T DL 17 | |
| | :L STD4 | Wert für Stunden x 10000 |
| | :T DR 17 | |
| | :UN M 10.2 | Stellwerte übernehmen |
| | :S M 10.2 | Bit 10 Statuswort MW 10 |
| | :S M 10.1 | Betriebsstd.zähler freigeben, falls noch nicht freigegeben. |
| | :L KT 020.1 | Überwachungszeit starten |
| | :SV T 12 | |
| M001 | :U T 12 | BEB, wenn Zeit noch |
| | :BEB | nicht abgelaufen |
| | :UN M 10.2 | wurden Stellwerte übern. |
| | :SPB =M002 | Sprung nach M002 |
| | :S =FEHL | Fehler-Bit setzen |
| | :BEA | |
| M002 | :UN M 10.0 | Fehler bei Stellwertvorg |
| | :RB =FEHL | Fehlerbit rücksetzen |
| | :BEB | BEB, wenn kein Fehler |
| | :S =FEHL | Fehler-Bit setzen |
| | :BE | |

**Beispiel: Lesen des Betriebsstunden-zählers**

Nach 300 Betriebsstunden soll eine Maschine ausgeschaltet werden, um eine Inspektion durchführen zu können. Merker 12.4 ist gesetzt, wenn die Maschine ausgeschaltet wurde. Nach Ablauf der 300 Betriebsstunden wird zum PB 5 verzweigt, der das Abschalten bewirken soll (im Beispiel nicht programmiert).

Der Uhrdatenbereich liegt im DB 2 ab MW 0, Statuswort ist MW 10.

OB 1

| OB 1 AWL | | Erläuterung |
|---|---|---|
| | : | **Betriebsstundenzähler auswerten** |
| | : | |
| | :SPA FB 14 | |
| NAME | :BETR-LES | |
| | : | |
| | :BE | |

FB 14

| FB 14 AWL | | Erläuterung |
|---|---|---|
| NAME | :BETRST-LES | **Betriebsstundenzähler lesen** |
| | : | |
| | :A DB 2 | DB, in dem Uhrdaten liegen |
| | :U M 12.4 | wenn HM 12.4 gesetzt, |
| | :BEB | ist Maschine schon ausgeschaltet-BEB |
| | :U M 10.1 | Betriebsstd.zähler sperren |
| | :R M 10.1 | (Bit 9 im Statuswort) |
| | :L DL 14 | Stundenwert x 100 in AKKU 1 laden |
| | :UN M 10.1 | Betriebsstd.zähler freigeben |
| | :S M 10.1 | |
| | :L KZ 003 | mit 3 (=300STD.) vergleichen |
| | :><F | |
| | :BEB | Ende, wenn 300 STD. noch nicht erreicht |
| | : | |
| | :S M 12.4 | HM setzen |
| | : | |
| | :SPA PB 5 | Wenn 300 Betriebsstd. |
| | : | erreicht sind, wir zum |
| | : | PB 5 verzweigt. |
| | : | |
| | :BE | |

# Teil 12 Alarm- und Zeitgesteuerte Bearbeitung

**Überblick** In diesem Kapitel erfahren Sie etwas über die Programmbearbeitung, bzw. die Unterbrechung des zyklischen Programmbetriebes mit der CPU 24x.

**Inhalt**


| Thema | Seite |
|---|---|
| **Teil 12 Alarm- und Zeitgesteuerte Bearbeitung** | **12-1** |
| Programmierung von Alarmbausteinen | 12-2 |
| Zeitgesteuerte Programmbearbeitung | 12-4 |
| Zeitdauergesteuerte Programmbearbeitung | 12-6 |
| Priorität und Reaktionszeit | 12-7 |
| Alarm-Diagnosedaten | 12-11 |

# Programmierung von Alarmbausteinen

**Grundlagen**

Im System 200V gibt es alarm-auslösende Baugruppen wie z.B. der Profibus-Master IM 208 von VIPA.

Die CPU ist ein alarmverarbeitender Teilnehmer. Sie empfängt Alarme und muss danach ihre Programmverarbeitung richten.

Der Alarm veranlasst das Betriebssystem der CPU, das zyklische oder das zeitgesteuerte Programm zu unterbrechen und einen Alarm-OB aufzurufen:

Die CPU 24x ruft bei einem Alarm den OB 2 auf.

Befindet sich die CPU im STOP-Zustand, so werden eintreffende Alarme ignoriert.

**Übersicht**

Neben der zyklischen Bearbeitung des Anwenderprogramms gibt es folgende asynchronen Programmbearbeitungen:

- Alarmgesteuerte Programmbearbeitung (OB 2),
- Zeitrastergesteuerte Programmbearbeitung (OB 10 ... OB 13)
- Zeitdauergesteuerte Programmbearbeitung (OB 6),
- Weckfehler (OB 33).

Sind die entsprechenden Bausteine nicht programmiert, so erfolgt keine Reaktion auf das Ereignis und das Programm wird unmittelbar nach dem Alarm fortgeführt. Treffen während der Alarmbearbeitung weitere Alarme ein (Flanke genügt!), dann wird ein Weckfehler ausgelöst.

**Auslösung**

Sobald ein Alarm den aktuellen Zyklus unterbrochen hat, d.h. während die CPU einen Alarm-OB bearbeitet, sind weitere Alarme gesperrt. Ein laufendes Alarm-Programm kann daher nicht unterbrochen werden.

Alarm hat grundsätzlich höhere Priorität als die "Zeitalarme" (OB10...OB13). Die Priorität des OB 6 gegenüber der Alarmbearbeitung ist einstellbar.

Ein Alarm unterbricht das zyklische und zeitgesteuerte Programm nach jeder Operation. Integrierte Funktionsbausteine und Betriebssystem-Routinen können durch einen Alarm nur an vorgegebenen Stellen unterbrochen werden (nicht beeinflussbar!)

**Programm vor Alarmunterbrechung schützen**

Wenn ein Teil Ihres zyklischen oder zeitgesteuerten Programms nicht unterbrochen werden darf, dann müssen Sie diesen Teil durch die Operation "AS" (Alarme sperren) vor Unterbrechungen schützen. Am Ende dieses "geschützten" Programmteils müssen Alarme durch die Operation "AF" wieder freigegeben werden.

Während der Alarm-Sperre kann ein Alarm gespeichert werden!

**Unterbrechungs-stellen**

Das zyklische Programm kann durch einen Interrupt nach jeder Anweisung unterbrochen werden.

Das Systemprogramm der CPU und die mitgelieferten Funktionsbausteine sind durch einen Interrupt an festgelegten Stellen unterbrechbar.

Die maximal mögliche Reaktionszeit ist abhängig von der gerade bearbeiteten Funktion.

Bei der CPU 24x kann ein Alarmprogramm weder durch einen weiteren Interrupt noch durch einen Zeitdaueralarm unterbrochen werden. Tritt ein neuer Interrupt während einer Alarmbearbeitung auf, wird ein Weckfehler ausgelöst.

Standardmäßig hat der OB 6 die höchste Bearbeitungspriorität.

**Hinweise für die Programmierung**

Beachten Sie, dass die Baustein-Schachtelungstiefe von 32 Ebenen auch bei dem Aufruf von Alarm-OBs nicht überschritten werden darf!

Wenn Sie im alarmverarbeitenden Programm die selben Merker wie im zyklischen Programm verwenden, müssen Sie den Inhalt dieser Merker zu Beginn des alarmverarbeitenden Programms z.B. in einem Datenbaustein retten. Am Ende des alarmverarbeitenden Programms transferieren Sie den geretteten Inhalt der Merker wieder zurück in die entsprechenden Merkerbytes (-worte).

**Merkerinhalte retten** ⇒ **Alarm bearbeiten** ⇒ **Merkerinhalte! zurückschreiben**

Bei der CPU 24x ist beim Ansprechen von kacheladressierten Baugruppen im Unterbrechungsprogramm die Kachelnummer ebenfalls zu retten und zu laden.

Die **Quittierung des Interrupts** erfolgt automatisch.

**Prozessabbild während eines Alarms**

Bei der Bearbeitung des Alarmprogramms werden die Prozessabbilder vom Systemprogramm nicht aktualisiert. Will man mit aktuellen Eingangswerten arbeiten oder will man im Alarmfall schnell über Digitalausgänge reagieren, müssen die Prozessabbilder im Alarmprogramm mit
L PB bzw. T PB aktualisiert werden.

**Alarme im Anlauf freigeben (OB 21, OB 22)**

Wenn Alarm-Reaktionen schon im Anlauf gewünscht sind, müssen Sie zu Beginn des Anlauf-OBs die Alarme mit der Operation "AF" gezielt freigeben. Andernfalls können Alarme erst nach Abarbeitung des Anlauf-OBs wirksam werden.

# Zeitgesteuerte Programmbearbeitung

**Übersicht**

Die zeitgesteuerten (auch zeitrastergesteuerte) Programmbearbeitung wird in einstellbaren Zeitintervallen ausgeführt. Für die zeitgesteuerte Programmbearbeitung stehen die Organisationsbausteine OB 10 bis OB 13 zur Verfügung.

| OBs | Systemdatenwort | Priorität |
|---|---|---|
| OB 13 | BS 97 | höchste |
| OB 12 | BS 98 | ↑ |
| OB 11 | BS 99 | |
| OB 10 | BS 100 | niedrigste |

Eine zeitgesteuerte Bearbeitung liegt vor, wenn ein von der internen Uhr kommendes Signal die CPU veranlasst, die zyklische Programmbearbeitung zu unterbrechen und ein spezifisches Programm (Zeitprogramm) zu bearbeiten.

Nach der Bearbeitung dieses Zeitprogramms setzt die CPU die Programmbearbeitung an der Unterbrechungsstelle fort.

Auf diese Weise kann ein bestimmtes Programm in einem von der Bearbeitungszeit des zyklischen Programms unabhängigen Zeitraster bearbeitet werden.

Treten an einer Unterbrechungsstelle "gleichzeitig" mehrere Weckalarme auf, hat der OB 13 die höchste Priorität (er wird zuerst bearbeitet), der OB 10 die niedrigste.

**Zeitraster vorgeben**

Die Festlegung der Zeitintervalle für das Zeitprogramm geschieht im Anlaufprogramm. Hierzu gibt es folgende 2 Möglichkeiten:

- In den Anlauf-Organisationsbausteinen OB 21 und OB 22 wird ein Funktionsbaustein aufgerufen, in welchem der entsprechende Wert zu den Systemdaten (BS 97 ... BS 100) transferiert wird.
  Bei der CPU 24x wird der Wert in den Systemdaten mit 10 ms multipliziert und entspricht dann dem Zeitraster.

  Das kleinste Zeitraster beträgt 10 ms und kann im Raster von 10 ms verändert werden.
- Die Zeitintervalle können auch im DB 1 unter der Blockkennung TFB: eingetragen werden.

**Beispiel: zeitgesteuerte Programm-bearbeitung**

Es soll der OB 10 im Raster von 10ms aufgerufen werden, der OB 11 im Sekundentakt.

Der Aufruf des OB 13 ist mit 100ms voreingestellt (Wert KF = 10 in BS 97). Der Aufruf des OB 12 ist standardmäßig gesperrt (Wert KF = 0 in BS 98).

*Einstellung durch Transfer in die Systemdaten*

```
L     KF    +1     Raster 10 ms für OB 10
T     BS    100
L     KF    +100   Raster 1s für OB 11
T     BS    99
```

*Einstellung über DB1 - Programmierung*

Die Zeitintervalle können auch im DB 1 unter der Blockkennung TFB: eingetragen werden.

```
KC =   TFB:          Blockkennung Zeitintervalle
C  =   OB10    10    Zeitintervall OB 10: 10ms
KC =   OB11    1000  Zeitintervall OB 11: 1s
C  =   ;             Blockendekennung
```

# Zeitdauergesteuerte Programmbearbeitung

**Funktion**

Die zeitdauergesteuerte Programmbearbeitung wird nach einer eingestellten Zeitdauer ausgeführt. Für die zeitdauergesteuerte Programmbearbeitung steht der OB 6 zur Verfügung.

Die Zeitdauer nach der der OB 6 aufgerufen werden soll können Sie über das Systemdatenwort BS 101 in Millisekunden vorgeben. Der Wert für die Zeitdauer kann im Bereich von 3 ms bis 65535 ms liegen.

Ist die Zeitdauer abgelaufen, trägt das Systemprogramm den Wert Null in das Systemdatum BS 101 ein und ruft den Organisationsbaustein OB 6 auf.

Wird während einer laufenden Zeitdauer die Anweisung T BS 101 erneut bearbeitet, wird erneut die Zeitdauer mit dem aktuell im Systemdatum BS 101 stehenden Wert gestartet. Wird der Wert Null in das Systemdatum BS 101 transferiert, wird eine laufende Zeitdauer gestoppt; der Aufruf des OB 6 wird dann verhindert.

**Beispiel: zeitdauergesteuerte Programmbearbeitung**

Der Organisationsbaustein OB 6 soll nach genau 135 ms aufgerufen werden. Das Programm lautet:

```
L     KF    +135
T     BS    101    Start der Zeitdauer
```

Die Anweisung T BS 101 kann nur in Funktionsbausteinen programmiert werden.

# Priorität und Reaktionszeit

In den vorherigen Abschnitten sind die möglichen Unterbrechungen des zyklischen Programms aus Gründen der Übersichtlichkeit so beschrieben, als würden sie einzeln auftreten. Wird nur eine Unterbrechungsart verwendet, genügt diese Beschreibung. Treten mehrere Unterbrechungen auf, hängt es von der Art und von der Bearbeitungspriorität der Unterbrechungen ab, wie diese nacheinander bearbeitet werden.

**Bearbeitungspriorität bei Unterbrechungen**

Folgende unterbrechungsgesteuerte Programmbearbeitungsarten können in das zyklische Programm eingeschoben werden:

- Zeitdauerbearbeitung aufgrund eines Zeitdaueralarms,
- Alarmbearbeitung aufgrund eines Interrupts
- Zeitbearbeitung aufgrund eines Weckalarms.

Treten alle Unterbrechungen "gleichzeitig" auf, werden sie standardmäßig in der oben angegebenen Reihenfolge bearbeitet, d.h. zuerst die Zeitdauerbearbeitung (OB 6), dann die Alarmbearbeitung (OB 2) und zum Schluss die Zeitbearbeitung (OB 10 bis OB 13). Innerhalb der Zeitbearbeitung wird der Organisationsbaustein OB 13 zuerst bearbeitet, dann der OB 12 usw..

Die Bearbeitungsreihenfolge des zeitdauergesteuerten Programms und des alarmgesteuerten Programms ist einstellbar. Standard wird die Zeitdauerbearbeitung vor der Alarmbearbeitung durchgeführt. Möchte man die Reihenfolge ändern, muss man das Systemdatenbit BS 120.6 setzen oder den Datenbaustein DB 1 entsprechend programmieren. In diesem Abschnitt ist die Bearbeitungsreihenfolge so beschrieben wie sie standardmäßig voreingestellt ist.

Ein Zeitdauerprogramm und ein Alarmprogramm sind nicht unterbrechbar. solange ein derartiges Programm läuft, werden die Unterbrechungsanforderungen gespeichert. Stehen nach der Bearbeitung des Zeitdauerprogramms und des Alarmprogramms Unterbrechungsanforderungen an, werden die entsprechenden Organisationsbausteine in der Reihenfolge der Bearbeitungspriorität aufgerufen.

Das zeitgesteuerte Programm ist durch den Zeitdaueralarm und durch Interrupts unterbrechbar. Nachdem das Zeitdauerprogramm und das Alarmprogramm bearbeitet worden sind, wird die Bearbeitung im Zeitprogramm fortgesetzt. Das zeitgesteuerte Programm ist durch Weckalarme nicht unterbrechbar. Treten Weckalarme während der Bearbeitung eines Zeitprogramms auf, ignoriert die CPU die Weckalarme.

**Reaktionszeiten bei Programm-unterbrechungen**

Die alarmgesteuerte Programmbearbeitung wendet man an, um auf ein Ereignis im Prozess schnell reagieren zu können, schneller als die Zyklusbearbeitungszeit es zulassen würde. Im Mittel liegt dann die Reaktionszeit auf einen Alarm oder einen Interrupt unter diesem Wert.

Eine wesentliche Rolle bei der Reaktionszeit auf eine Unterbrechung spielt das Systemprogramm. Arbeitet der Zentralprozessor gerade im Systemprogramm und führt er bestimmte Funktionen aus (z.B. Online-Funktionen für ein angeschlossenes Programmiergerät), kann sich die Reaktionszeit wesentlich vergrößern. Die Reaktionszeit auf eine Unterbrechungsanforderung im Systemprogramm setzt sich aus einer Grundreaktionszeit und einer Zusatzreaktionszeit zusammen, die von der gerade bearbeiteten Funktion abhängig ist.

**Interrupts**

Da das Programm nach jeder Anweisung unterbrochen werden kann, ist normalerweise eine sofortige Reaktion auf einen Interrupt gegeben.

Die am längsten laufende Anweisung ist ein Blocktransfer TNB bzw. TNW.

Größere Auswirkungen auf die Reaktionszeit hat in der Regel eine Sperre der Unterbrechungsbearbeitung, da diese Sperre über mehrere Anweisungen besteht. Diese Zeit ist in der Regel die maßgebende Reaktionszeit im Anwenderprogramm auf einen Interrupt.

Bei der CPU 24x kann ein Interruptprogramm durch ein anderes nicht unterbrochen werden; deshalb kann ein kurz nach einem Interrupt ausgelöster zweiter Interrupt erst dann bearbeitet werden, wenn das bereits laufende Interruptprogramm abgearbeitet ist. Somit kann die Bearbeitungszeit des am längsten laufenden Interruptprogramms auch in die Reaktionszeit mit eingehen.

Das eben Beschriebene gilt auch für ein Zeitdauerprogramm. Auch ein Zeitdauerprogramm im Organisationsbaustein OB 6 ist durch einen Interrupt nicht unterbrechbar, unabhängig von der eingestellten Bearbeitungspriorität. Unter Umständen muss man also auch die Bearbeitungszeit eines Zeitdauerprogramms bei der Betrachtung der Interrupt-Reaktionszeit mit berücksichtigen.

**Zeitdaueralarm**

Bei der CPU 24x kann ein Alarmprogramm nicht unterbrochen werden, so dass ein Zeitdaueralarm erst nach einem Alarmprogramm bearbeitet werden kann. Es kann jedoch die Bearbeitungspriorität des Zeitdaueralarms vor die der Interrupts gelegt werden, so dass bei einem "gleichzeitigen" Auftreten von Interrupts und einem Zeitdaueralarm der Zeitdaueralarm vor dem Interrupts bearbeitet wird.

**Weckalarme**

Bei Weckalarmen ist die Reaktionszeit in den meisten Fällen nicht so bedeutsam wie bei den Interrupts. Wird das zu einem Weckalarm gehörende Programm verzögert zum Auftreten des Weckalarms bearbeitet, verschiebt sich deshalb nicht der Grundtakt. Bei der nächsten Zeitperiode tritt der gleiche Weckalarm exakt im eingestellten Zeitraster auf, so dass sich im Mittel eine gute zeitliche Konstanz in der Bearbeitung ergibt. Der prozentuale zeitliche Fehler der verzögerten Bearbeitungszeit ist naturgemäß bei den kurzen Zeitperioden größer als bei den langen.

Das Systemprogramm und die integrierten Funktionsbausteine sind durch Weckalarme nicht unterbrechbar. Bei laufenden Systemfunktionen kann zum eingestellten Zeitintervall noch die Summe aus der Grundreaktionszeit des Systemprogramms und die Reaktionszeit der ausgeführten Systemfunktionen hinzukommen.

Eine laufende Alarmbearbeitung oder eine laufende Zeitdauerbearbeitung sind nicht unterbrechbar. Zusätzlich haben die Interrupts und der Zeitdaueralarm eine höhere Bearbeitungspriorität als die Weckalarme, so dass die Bearbeitungszeiten des Alarmprogramms und des Zeitdauerprogramms eine Rolle bei der Reaktion auf einen Weckalarm spielen können.

Zwischen den Anweisungen *Alarme sperren "AS"* und *Alarme freigeben* "*AF"* werden keine Weckalarme bearbeitet. Die Zeitdauer dieser Sperre kann also auch in die Betrachtung der Weckalarm-Reaktionszeit mit eingehen.

Bei der CPU 24x ist ein Zeitprogramm durch einen Weckalarm nicht unterbrechbar. Deshalb ist die Laufzeit des am längsten laufenden Zeitprogramms mit in die Überlegung der Reaktionszeit einzubeziehen.

**Weckfehler**

Ein Weckfehler wird gemeldet, wenn ein zeitdauergesteuertes oder zeitgesteuertes Programm bearbeitet und die Bearbeitung ein zweites Mal angefordert wird.

Ein Weckfehler wird ebenfalls ausgelöst falls die Bearbeitung während einer Alarmsperre ein zweites Mal angefordert wird. Ein Weckfehler hat den Aufruf des OB 33 zur Folge. Im AKKU 1 wird ein Fehlercode übergeben, der den Weckfehler näher spezifiziert:

| Fehlercode in AKKU 1 | Ursache |
|---|---|
| 1 | Bearbeitung des OB 6 |
| 2 | Bearbeitung des OB 13 |
| 3 | Bearbeitung des OB 12 |
| 4 | Bearbeitung des OB 11 |
| 5 | Bearbeitung des OB 10 |

**Reaktionszeiten**

Die Reaktionszeit des zyklisch bearbeiteten Programms ist maßgebend abhängig von der Zyklusbearbeitungszeit.

Jede Unterbrechung der zyklischen Programmbearbeitung (Interrupts, Weckalarme und Zeitdaueralarm) verlängern durch deren Bearbeitung die Zeitdauer zwischen der Übertragung der Prozessabbilder und damit die Reaktionszeit des zyklischen Programms.

Wird das zyklische Programm sehr häufig unterbrochen, kann es sogar vorkommen, dass die Zyklusüberwachungszeit anspricht und "Zykluszeit-überschreitung" meldet.

# Alarm-Diagnosedaten

Wird ein Prozess- oder Diagnosealarm gemeldet, so erfolgt der Aufruf des OB 2. Um dem Anwender die Möglichkeit zu geben mehrere alarmfähige Baugruppen in einem System zu verwenden, ist eine Alarmdiagnose eingeführt worden.

Bei jedem Alarm werden im BS-Bereich Diagnosedaten abgelegt, die die Alarmursache und den Alarmverursacher näher spezifizieren.

Es sind maximal 9 Datenworte für Diagnosedaten vorgesehen und sie haben folgenden Aufbau:

*Diagnosedaten im BS-Bereich*

| BS-Wort | Inhalt | | |
|---|---|---|---|
| BS 110 Low | Adresse der Baugruppe (=Steckplatz 1...32) | | |
| BS 110 High | Alarmkennung:<br>1h=Prozessalarm<br>2h=Diagnosealarm<br>3h=Hardwarefehler | | |
| | IM 208 Slaveausfall | IM 208 Run → Stop | Hardwarefehler<br>(BS 110 High = 3): |
| BS 111 Low | 01h=Diagnose<br>Slaveausfall | 04h=Diagnosestop | interne Fehlerdaten |
| BS 111 High | 00h... 7Fh=<br>Stationsadresse | | interne Fehlerdaten |
| BS 112 | | | interne Fehlerdaten |
| BS 113 | | | interne Fehlerdaten |
| BS 114 | | | interne Fehlerdaten |

**Verlust von Alarmen**

Falls ein oder mehrere Prozess- oder Diagnosealarme bei laufender Alarmbearbeitung gemeldet werden, so führt dies zu einem Weckfehler. Der Weckfehler hat wie üblich den Aufruf des OB 33 zur Folge.

Die in AKKU 1 übergebenen Fehlercodes werden entsprechend der Alarmkennung erweitert.

| Fehlercode in AKKU 1 | Ursache |
|---|---|
| 6 | Bearbeitung Alarm |

Ab BS-Zellen 110 ff. werden die Daten des Alarms vom niedrigsten Steckplatz eingetragen.

**Zeitbedarf für die Alarmbearbeitung**

Jeder Alarm hat in der CPU einen Interrupt zur Folge. Dieser Interrupt leitet die Alarmbearbeitung ein, indem ein Systemflag gesetzt wird. Das Systemflag wird an der nächsten Befehlsgrenze im Interpreter ausgewertet. Wann der nächste Befehl im Interpreter abgearbeitet wird, hängt aber davon ab welche Aktivitäten aktuell in der CPU durchgeführt werden. Im Idealfall ist die CPU mit der Abarbeitung des Interpreters im Zyklus beschäftigt und somit verzögert sich die Abarbeitung des Alarms um maximal einen Befehl. Der Alarm kann aber ebenso außerhalb der Zyklusbearbeitung oder während der Bearbeitung einer seriellen Kommunikation auftreten. Hierbei können Verzögerungen in der Größenordnung von 100ms auftreten.

Der schlechteste Fall kann nur durch genauere Untersuchung ermittelt werden. Der komplette Zeitbedarf einer Alarmbearbeitung wird des weiteren von der programmierten Reaktion im OB 2 bestimmt. Alarmsperren und eine zeitdauergesteuerte Bearbeitung gehen ebenfalls zu Lasten der Alarmbearbeitungszeit.

# Anhang

## A Befehlsliste

| Abkürzungen | | Erläuterungen |
|---|---|---|
| AKKU 1 | | Akkumulator 1 (Beim Laden des AKKU 1 wird der ursprüngliche Inhalt in den AKKU 2 geschoben) |
| AKKU 2 | | Akkumulator 2 |
| ANZ 0/ANZ 1 | | Ergebnisanzeige 0/Ergebnisanzeige 1 |
| AWL | | Darstellungsart Anweisungsliste |
| FOOP | | Formaloperand, Ausdruck mit max. 4 Zeichen, wobei das erste Zeichen ein Buchstabe sein muss. |
| FUP | | STEP 5-Darstellungsart Funktionsplan (von Siemens) |
| KOP | | STEP 5-Darstellungsart Kontaktplan (von Siemens) |
| MARK | | Symbolische Adresse mit maximal 4 Zeichen |
| OV | | Überlauf-Anzeige (Overflow). Wird bei überschrittenem Zahlenbereich gesetzt |
| PAE | | Prozessabbild der Eingänge |
| PAA | | Prozessabbild der Ausgänge |
| VKE | | Verknüpfungsergebnis |
| VKE abh. | J | Die Anweisung wird nur ausgeführt, wenn das VKE = “1“ ist. |
| | J ↑/↓ | Die Anweisung wird nur ausgeführt, wenn positiver/negativer Flankenwechsel beim VKE vorliegt. |
| | N | Die Anweisung wird immer ausgeführt. |
| VKE beeinfl. | J/N | Das VKE wird durch die Operation beeinflusst/nicht beeinflusst. |
| | 1 | VKE wird gleich "1" gesetzt. |
| VKE begr. | J | Das VKE wird nicht verändert. Eine Weiterverknüpfung ist nicht mehr möglich. |
| | N | Bei der nächsten Binärverknüpfung (aber nicht Zuweisung) wird das VKE neu aufgebaut. Abhängig davon, ob die Operation VKE beeinflussend ist oder nicht, wird das VKE entsprechend der Operation und des Status des abgefragten Bits weiter verknüpft oder unverändert belassen. |

| nur FB | Befehl | Operand | Parameter | VKE 1 abh. 2 beeinfl 3 begr. | | | Anz. | | | Funktionsbeschreibung | Befehls-Code (hex) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 | 1 | 0 | OV | | |
| | | | | **Arithmetische Operationen** | | | | | | | |
| | +F | | | N | N | N | x | x | x | Zwei Festpunktzahlen addieren: AKKU 2 + AKKU 1 = AKKU 1 | 7900 |
| | -F | | | N | N | N | x | x | x | Zwei Festpunktzahlen subtrahieren: AKKU 2 - AKKU 1 = AKKU 1 | 5900 |
| x | D | | 0 ... 255 | N | N | N | | | | Low-Byte (Bits 0-7) von AKKU 1 um den Wert n (n=0...255) dekrementieren (kein Übertrag in High-Byte von AKKU 1) | $190_w0_w$ |
| x | I | | 0 ... 255 | N | N | N | | | | Low-Byte (Bits 0-7) von AKKU 1 um den Wert n (n=0...255) inkrementieren (kein Übertrag in High-Byte von AKKU 1) | $110_w0_w$ |
| x | ADD | BF | -128 ... +127 | N | N | N | | | | Byte-Konstante (Festpunkt) zum AKKU 1 addieren. | $500_k0_k$ |
| x | | KF | -32768 ... +32767 | N | N | N | | | | Wort-Konstante (Festpunkt) zum AKKU 1 addieren. | 5800 $0_k0_k0_k0_k$ |
| | | | | **Baustein-Operationen** | | | | | | | |
| | A | DB | 0 ... 255 | N | N | N | | | | Einen Datenbaustein öffnen | $200_n0_n$ |
| | E | DB | 0 ... 255 | N | N | N | | | | Einen Datenbaustein erzeugen mit Länge in AKKU 1 | 7805 $000_n0_n$ |
| | SPA | OB | 0 ... 255 | N | N | J | | | | Absolut (unbedingt) zu einem OB springen. | $6D0_n0_n$ |
| | | PB | 0 ... 255 | N | N | J | | | | Absolut (unbedingt) zu einem PB springen. | $750_n0_n$ |
| | | FB | 0 ... 255 | N | N | J | | | | Absolut (unbedingt) zu einem FB springen. | $3D0_n0_n$ |
| | | SB | 0 ... 255 | N | N | J | | | | Absolut (unbedingt) zu einem SB springen. | $7D0_n0_n$ |
| | SPB | OB | 0 ... 255 | J | 1 | J | | | | Bedingt (wenn VKE=1) zu einem OB springen. | $4D0_n0_n$ |
| | | PB | 0 ... 255 | J | 1 | J | | | | Bedingt (wenn VKE=1) zu einem PB springen. | $550_n0_n$ |
| | | FB | 0 ... 255 | J | 1 | J | | | | Bedingt (wenn VKE=1) zu einem FB springen. | $1D0_n0_n$ |
| | | SB | 0 ... 255 | J | 1 | J | | | | Bedingt (wenn VKE=1) zu einem SB springen. | $5D0_n0_n$ |
| | BE | | | N | N | J | | | | Bausteinende (letzter Befehl) | 6500 |
| | BEA | | | N | N | J | | | | Bausteinende (wie BE, jedoch nicht der letzte Befehl) | 6501 |
| | BEB | | | J | 1 | J | | | | Bausteinende bedingt | 0500 |
| x | B | =FOOP | | N | N | N | | | | Baustein bearbeiten. (Nur A DB, SPA, PB, SPA FB, SPA SB und SPA OB) | $760_p0_p$ |
| | | | | **Bildaufbau- und Null-Operationen** | | | | | | | |
| | BLD | | 130 | N | N | N | | | | Bildaufbau-Befehl für das PG: Leerzeile | 1082 |
| | | | 131 | N | N | N | | | | Bildaufbau-Befehl für das PG: Umschalten auf AWL | 1083 |
| | | | 132 | N | N | N | | | | Bildaufbau-Befehl für das PG: Umschalten auf FUP | 1084 |
| | | | 133 | N | N | N | | | | Bildaufbau-Befehl für das PG: Umschalten auf KOP | 1085 |
| | | | 255 | N | N | N | | | | Bildaufbau-Befehl für das PG: Segment Ende | 10FF |
| | NOP | 0 | | N | N | N | | | | Nulloperation (alle Bits gelöscht) | 0000 |
| | | 1 | | N | N | N | | | | Nulloperation (alle Bits gesetzt) | FFFF |
| | | | | **Bit-Test-Operationen** | | | | | | | |
| x | P | T | 0.0 ... 127.15 | N | J | N | | | | Bit eines Zeitwortes auf Signalzustand "1" prüfen. | 7025 $C00_t0_t$ |
| x | | Z | 0.0 ... 127.15 | N | J | N | | | | Bit eines Zählerwortes auf Signalzustand "1" prüfen. | 7015 $C00_z0_z$ |
| x | | D | 0.0 ... 255.15 | N | J | N | | | | Bit eines Datenwortes auf Signalzustand "1" prüfen. | 7046 $C0_i0_o0_o$ |
| x | | BS | 0.0 ... 255.15 | N | J | N | | | | Bit eines Systemdatums auf Signalzustand "1" prüfen. | 7057 $C0_i0_o0_o$ |

| nur FB | Befehl | Operand | Parameter | VKE 1 abh. 2 beeinfl 3 begr. | | | Anz. | | | Funktionsbeschreibung | Befehls-Code (hex) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 | 1 | 0 | OV | | |
| x | PN | T | 0.0 ... 127.15 | N | J | N | | | | Bit eines Zeitwortes auf Signalzustand "0" prüfen. | 7025 $800_t0_t$ |
| x | | Z | 0.0 ... 127.15 | N | J | N | | | | Bit eines Zählerwortes auf Signalzustand "0" prüfen. | 7015 $800_z0_z$ |
| x | | D | 0.0 ... 255.15 | N | J | N | | | | Bit eines Datenwortes auf Signalzustand "0" prüfen. | 7046 $80_i0_o0_o$ |
| x | | BS | 0.0 ... 255.15 | N | J | N | | | | Bit eines Systemdatums auf Signalzustand "0" prüfen. | 7057 $80_i0_o0_o$ |
| | | | | | | | | | | **Lade-Operationen mit Operanden** | |
| | L | EB | 0 ... 127 | N | N | N | | | | Eingangsbyte vom PAE in den AKKU 1 laden. | $4A0_y0_y$ |
| | | EW | 0 ... 126 | N | N | N | | | | Eingangswort vom PAE in den AKKU 1 laden: Byte n→AKKU 1 (Bits 8-15); Byte n+1→AKKU 1 (Bits 0-7). | $520_y0_y$ |
| | | PB/PY | 0 ... 255 | N | N | N | | | | Peripheriebyte der Digital-/Analog-Eingabe in AKKU 1 laden. | $720_y0_y$ |
| | | PW | 0 ... 254 | N | N | N | | | | Peripheriewort der Digital-/Analog-Eingabe in AKKU 1 laden. Byte n→AKKU 1 (Bits 8-15); Byte n+1→AKKU 1 (Bits 0-7). | $7A0_y0_y$ |
| | | AB | 0 ... 127 | N | N | N | | | | Ausgangsbyte vom PAA in AKKU 1 laden. | $4A8_y0_y$ |
| | | AW | 0 ... 126 | N | N | N | | | | Ausgangswort vom PAA in AKKU 1 laden: Byte n→AKKU 1 (Bits 8-15); Byte n+1→AKKU 1 (Bits 0-7). | $528_y0_y$ |
| | | MB | 0 ... 255 | N | N | N | | | | Merkerbyte in AKKU 1 laden. | $0A0_y0_y$ |
| | | MW | 0 ... 254 | N | N | N | | | | Merkerwort in AKKU 1 laden. Byte n→AKKU 1 (Bits 8-15); Byte n+1→AKKU 1 (Bits 0-7). | $120_y0_y$ |
| | | SY | 0 ...1023 | N | N | N | | | | S-Merkerbyte in AKKU 1 laden | 78AB |
| | | SW | 0 ...1022 | N | N | N | | | | S-Merkerwort in AKKU 1 laden Byte n→AKKU 1 (Bits 8-15); Byte n+1→AKKU 1 (Bits 0-7). | 78CB |
| | | DW | 0 ... 255 | N | N | N | | | | Datenwort des aktuellen DB in AKKU 1 laden | $320_o0_o$ |
| | | DL | 0 ... 255 | N | N | N | | | | Daten (linkes Byte) des aktuellen DBs in AKKU 1 laden. | $220_o0_o$ |
| | | DR | 0 ... 255 | N | N | N | | | | Daten (rechtes Byte) des aktuellen DBs in AKKU 1 laden. | $2A0_o0_o$ |
| | | T | 0 ... 127 | N | N | N | | | | Einen Zeitwert (dual-codiert) in AKKU 1 laden. | $020_d0_d$ |
| | | Z | 0 ... 127 | N | N | N | | | | Einen Zählwert (dual-codiert) in AKKU 1 laden. | $420_z0_z$ |
| x | | BS | 0 ... 255 | N | N | N | | | | Ein Wort aus dem Bereich Systemdaten in AKKU 1 laden. | $620_o0_o$ |
| x | | =FOOP | | N | N | N | | | | Den Wert des Formaloperanden in AKKU 1 laden. (Parametertyp: BY, W; Aktualop.: E, A, M, DL, DR, DW) | $460_p0_p$ |
| x | LIR | | 0(→AKKU 1) 2(→AKKU 2) | N | N | N | | | | AKKU 1 oder 2 mit dem Inhalt eines Speicherwortes (durch AKKU 1 adressiert) indirekt laden (0=AKKU 1, 2=AKKU 2) | $4000_r$ |
| x | LDI | | A1(→AKKU 1) | N | N | N | | | | AKKU 1 mit dem Inhalt eines Speicherwortes (durch AKKU 1 adressiert) indirekt laden (A1=AKKU 1) | 680B |
| | | | A2(→AKKU 2) | N | N | N | | | | AKKU 2 mit dem Inhalt eines Speicherwortes (durch AKKU 1 adressiert) indirekt laden (A2=AKKU 2) | 682B |
| | LC | T | 0 ... 127 | N | N | N | | | | Zeitwerte (BCD-codiert) in den AKKU 1 laden. | $0C0_t0_t$ |
| | | Z | 0 ... 127 | N | N | N | | | | Zählwerte (BCD-codiert) in den AKKU 1 laden. | $4C00_z$ |
| x | | =FOOP | | N | N | N | | | | Wert des Formaloperanden im BCD-Code in den AKKU 1 laden. (Aktualoperand: T, Z) | $0E0_p0_p$ |
| x | LW | =FOOP | | N | N | N | | | | Bitmuster eines Formaloperanden in den AKKU 1 laden (Parameterart: D; Parametertyp: KF, KH, KM, KY, KC, KT, KZ) | $3F0_p0_p$ |
| | | | | | | | | | | **Lade-Operationen mit Konstanten** | |
| | L | KB | 0 ... 255 | N | N | N | | | | Konstante (1-Byte-Zahl) in den AKKU 1 laden. | $280_k0_k$ |
| | | KC | Character | N | N | N | | | | Konstante (2 Character-Zeichen) in AKKU 1 laden. | 3010 $0_k0_k0_k0_k$ |
| | | KF | -32768 ... +32767 | N | N | N | | | | Konstante (Festpunktzahl) in den AKKU 1 laden. | 3004 $0_k0_k0_k0_k$ |
| | | KH | 0000 ... FFFF | N | N | N | | | | Konstante (Hexa-Code) in den AKKU 1 laden. | 3040 $0_k0_k0_k0_k$ |
| | | KM | Bitmuster (16 Bit) | N | N | N | | | | Konstante (Bitmuster) in den AKKU 1 laden. | 3080 $0_k0_k0_k0_k$ |

| nur FB | Befehl | Operand | Parameter | VKE 1 abh. 2 beeinfl 3 begr. 1 | 2 | 3 | Anz. 1 | 0 | OV | Funktionsbeschreibung | Befehls-Code (hex) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | KY | 0 ... 255, 0... 255 | N | N | N | | | | Konstante (2-Byte-Zahl) in den AKKU 1 laden. | 3020 $0_k0_k0_k0_k$ |
| | | KT | 0.0 ... 999.3 | N | N | N | | | | Konstante (Zeitwert) in den AKKU 1 laden | 3002 $0_k0_k0_k0_k$ |
| | | KZ | 0 ... 999 | N | N | N | | | | Konstante (Zählwert) in den AKKU 1 laden | 3001 $0_k0_k0_k0_k$ |
| | | | | | | | | | | **Schiebe-Operationen (digital)** | |
| x | SLW | | 0 ... 15 | N | N | N | x | x | | Inhalt von AKKU 1 um den angegebenen Wert nach links schieben. Freiwerdende Stellen werden mit Nullen aufgefüllt. | $610_s0_s$ |
| x | SRW | | 0 ... 15 | N | N | N | x | x | | Inhalt von AKKU 1 um den angegebenen Wert nach rechts schieben. Freiwerdende Stellen werden mit Nullen aufgefüllt. | $690_s0_s$ |
| | | | | | | | | | | **Setzoperationen** | |
| | S | E | 0.0 ... 127.7 | J | N | J | | | | Den Operanden bei VKE = 1 auf den Wert "1" setzen. | $D0_i0_y0_y$ |
| | | A | 0.0 ... 127.7 | J | N | J | | | | Den Operanden bei VKE = 1 auf den Wert "1" setzen. | $D0_i8_y0_y$ |
| | | M | 0.0 ... 255.7 | J | N | J | | | | Den Operanden bei VKE = 1 auf den Wert "1" setzen. | $90_i0_y0_y$ |
| | | S | 0.0 ... 1023.7 | J | N | J | | | | Den Operanden bei VKE = 1 auf den Wert "1" setzen. | 782B |
| | R | E | 0.0 ... 127.7 | J | N | J | | | | Den Operanden bei VKE = 1 auf den Wert "0" rücksetzen. | $F0_i0_y0_y$ |
| | | A | 0.0 ... 127.7 | J | N | J | | | | Den Operanden bei VKE = 1 auf den Wert "0" rücksetzen. | $F0_i8_y0_y$ |
| | | M | 0.0 ... 255.7 | J | N | J | | | | Den Operanden bei VKE = 1 auf den Wert "0" rücksetzen. | $B0_i0_y0_y$ |
| | | S | 0.0 ... 1023.7 | J | N | J | | | | Den Operanden bei VKE = 1 auf den Wert "0" rücksetzen. | 786B |
| x | SU | T | 0 ... 127 | N | N | J | | | | Bit eines Zeitwortes unbedingt setzen. | 7025 $400_t0_t$ |
| x | | Z | 0 ... 127 | N | N | J | | | | Bit eines Zählerwortes unbedingt setzen. | 7015 $400_z0_z$ |
| x | | D | 0.0 ... 255.15 | N | N | J | | | | Bit eines Datenwortes unbedingt setzen. | 7046 $40_i0_o0_o$ |
| x | | BS | 0.0 ... 255.15 | N | N | J | | | | Bit im Bereich der Systemdaten unbedingt setzen. | 7057 $40_i0_o0_o$ |
| x | RU | T | 0 ... 127 | N | N | J | | | | Bit eines Zeitwortes unbedingt rücksetzen. | 7025 $000_t0_t$ |
| x | | Z | 0 ... 127 | N | N | J | | | | Bit eines Zählerwortes unbedingt rücksetzen. | 7015 $000_z0_z$ |
| x | | D | 0.0 ... 255.15 | N | N | J | | | | Bit eines Datenwortes unbedingt rücksetzen. | 7046 $00_i0_o0_o$ |
| x | | BS | 0.0 ... 255.15 | N | N | J | | | | Bit im Bereich der Systemdaten unbedingt rücksetzen. | 7057 $00_i0_o0_o$ |
| x | S | =FOOP | | J | N | J | | | | Einen Formaloperanden bei VKE = 1 setzen (Parametertyp: BI, Aktualoperanden: E, A, M) | $170_p0_p$ |
| x | RB | =FOOP | | J | N | J | | | | Einen Formaloperanden bei VKE = 1 rücksetzen (Parametertyp: BI, Aktualoperanden: E, A, M) | $370_p0_p$ |
| x | RD | =FOOP | | J | N | J | | | | Einen Formaloperanden bei VKE = 1 rücksetzen digital (Parametertyp: BY, Aktualoperanden: T, Z) | $3E0_p0_p$ |
| | | | | | | | | | | **Speicheroperationen** | |
| | = | E | 0.0 ... 127.7 | J | N | J | | | | Dem Operanden wird der Wert des VKE zugewiesen. | $D8_i0_y0_y$ |
| | | A | 0.0 ... 127.7 | J | N | J | | | | Dem Operanden wird der Wert des VKE zugewiesen. | $D8_i8_y0_y$ |
| | | M | 0.0 ... 255.7 | J | N | J | | | | Dem Operanden wird der Wert des VKE zugewiesen. | $98_i0_y0_y$ |
| | | S | 0.0 ... 1023.7 | J | N | J | | | | Dem Operanden wird der Wert des VKE zugewiesen. | 783B |
| x | | =FOOP | | J | N | J | | | | Status des Formaloperanden den Wert des VKE zugeweisen. (Parametertyp: BI, Aktualoperanden: E, A, M) | $1F0_p0_p$ |
| | | | | | | | | | | **Sprung-Operationen** | |

| nur FB | Befehl | Operand | Parameter | VKE 1 abh. 2 beeinfl 3 begr. | | | Anz. | | | Funktionsbeschreibung | Befehls-Code (hex) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 | 1 | 0 | OV | | |
| x | SPA | =MARK | | N | N | N | | | | Absolut (unbedingt) zur Symboladresse springen. | $2D0_m0_m$ |
| x | SPB | =MARK | | J | 1 | J | | | | Bedingter Sprung zu Symboladresse (VKE = "0" → VKE = "1") | $FA0_m0_m$ |
| x | SPZ | =MARK | | N | N | N | | | | Sprung bei Ergebnis = 0: nur wenn ANZ 1 = 0 und ANZ 0 = 0. | $450_m0_m$ |
| x | SPN | =MARK | | N | N | N | | | | Sprung bei Ergebnis ≠ 0: nur wenn ANZ 1 ≠ ANZ 0 | $350_m0_m$ |
| x | SPP | =MARK | | N | N | N | | | | Sprung bei Ergebnis > 0: nur wenn ANZ 1 = 1 und ANZ 0 = 0 | $150_m0_m$ |
| x | SPM | =MARK | | N | N | N | | | | Sprung bei Ergebnis < 0: nur wenn ANZ 1 = 0 und ANZ 0 = 1 | $250_m0_m$ |
| x | SPO | =MARK | | N | N | N | | | | Sprung bei "Überlauf": nur wenn "OVERFLOW" gesetzt ist | $0D0_m0_m$ |
| x | SPR | | -32768 ... +32767 | N | N | N | | | | Relativer Sprung innerhalb eines FBs | $700B\ 0_j0_j0_j0_j$ |
| | | | | | | | | | | **Stop-Operationen** | |
| | STP | | | N | N | N | | | | Stop: Zyklus wird noch beendet. STS im USTACK wird gesetzt. (Stopanmeldung) | 7003 |
| x | STS | | | N | N | N | | | | Stop-Befehl: unmittelbar nach dem Befehl wird die Programmbearbeitung abgebrochen. | 7000 |
| | | | | | | | | | | **Transfer-Operationen** | |
| | T | EB | 0 ... 127 | N | N | N | | | | Inhalt AKKU 1 zu einem Eingangsbyte transferieren (ins PAE) | $4B0_y0_y$ |
| | | AB | 0 ... 127 | N | N | N | | | | Inhalt AKKU 1 zu einem Ausgangsbyte transferieren (ins PAA) | $4B8_y0_y$ |
| | | EW | 0 ... 126 | N | N | N | | | | Inhalt AKKU 1 zu einem Eingangswort transferieren (ins PAE): AKKU 1 (Bits 8-15)→Byte n; AKKU 1 (Bits 0-7)→Byte n+1 | $530_y0_y$ |
| | | AW | 0 ... 126 | N | N | N | | | | Inhalt AKKU 1 zu Ausgangswort transferieren (ins PAA): AKKU 1 (Bits 8-15)→Byte n; AKKU 1 (Bits 0-7)→Byte n+1 | $538_y0_y$ |
| | | PB/PY | 128 ... 255 | N | N | N | | | | Inhalt AKKU 1 zu Peripheriebyte der Digital-Ausgaben mit Nachführen des PAA oder der Analog-Ausgaben transferieren | $730_y0_y$ |
| | | PW | 0 ... 254 | N | N | N | | | | Inhalt AKKU 1 zu Peripheriewort der Digital-Ausgaben mit Nachführen des PAA oder der Analog-Ausgaben transferieren | $7B0_y0_y$ |
| | | MB | 0 ... 255 | N | N | N | | | | Inhalt AKKU 1 zu einem Merkerbyte transferieren. | $0B0_y0_y$ |
| | | MW | 0 ... 254 | N | N | N | | | | Inhalt AKKU 1 zu einem Merkerwort transferieren (ins PAA): AKKU 1 (Bits 8-15)→Byte n; AKKU 1 (Bits 0-7)→Byte n+1. | $130_y0_y$ |
| | | SY | 0 ... 1023 | N | N | N | | | | Inhalt von AKKU 1 zu einem S-Merkerbyte transferieren | 78BB |
| | | SW | 0 ... 1022 | N | N | N | | | | Inhalt von AKKU 1 zu einem S-Merkerwort transferieren | 78DB |
| | | DL | 0 ... 255 | N | N | N | | | | Inhalt AKKU 1 zu einem Datenwort (linkes Byte) des aktuellen Datenbausteins transferieren. | $230_o0_o$ |
| | | DR | 0 ... 255 | N | N | N | | | | Inhalt AKKU 1 zu einem Datenwort (rechtes Byte) des aktuellen Datenbausteins transferieren. | $2B0_o0_o$ |
| | | DW | 0 ... 255 | N | N | N | | | | Inhalt AKKU 1 zu einem Datenwort des aktuellen Datenbausteins transferieren. | $330_o0_o$ |
| x | | BS | 0 ... 255 | N | N | N | | | | Ein Wort in den Bereich der Systemdaten transferieren | $630_o0_o$ |
| x | | =FOOP | | N | N | N | | | | Inhalt AKKU 1 zum Formaloperanden transferieren (Parametertyp: BY, W; Aktualoperanden: E, A, M, DL, DR, DW) | $660_p0_p$ |
| x | TIR | | 0(→AKKU 1) 2(→AKKU 2) | N | N | N | | | | Inhalt des Akkumulators (0=AKKU 1, 2=AKKU 2) in das Speicherwort (durch AKKU 1 adressiert) indirekt transferieren. | $4800_r$ |
| x | TDI | | A1(→AKKU 1) | N | N | N | | | | Inhalt des Akkumulators (A1=AKKU 1) in das Speicherwort (durch AKKU 1 adressiert) indirekt transferieren. | 680F |
| | | | A2(→AKKU 2) | N | N | N | | | | Inhalt des Akkumulators (A2=AKKU 2) in das Speicherwort (durch AKKU 1 adressiert) indirekt transferieren. | 682F |
| x | TNB | | 0 ... 255 | N | N | N | | | | Byteweiser Blocktransfer (Anzahl der Bytes 0 ... 255) Endadresse Quelle: AKKU 2, Endadresse Ziel: AKKU 1 | $030_l0_l$ |
| | | | | | | | | | | **Umwandlungs-Operationen (digital)** | |
| x | KEW | | | N | N | N | | | | 1-er-Komplement von AKKU 1 bilden. | 0100 |
| x | KZW | | | N | N | N | x | x | x | 2-er-Komplement von AKKU 1 bilden. | 0900 |

| nur FB | Befehl | Operand | Parameter | VKE 1 abh. 2 beeinfl 3 begr. | | | Anz. | | | Funktionsbeschreibung | Befehls-Code (hex) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 | 1 | 0 | OV | | |
| | | | | | | | | | | **Vergleichs-Operationen (digital)** | |
| | ! = F | | | N | J | N | x | x | | Vergleich zweier Festpunktzahlen auf gleich:<br>AKKU 2 = AKKU 1 → VKE = "1". | 2180 |
| | > < F | | | N | J | N | x | x | | Vergleich zweier Festpunktzahlen auf ungleich:<br>AKKU 2 ≠ AKKU 1 → VKE = "1". | 2160 |
| | > F | | | N | J | N | x | x | | Vergleich zweier Festpunktzahlen auf größer:<br>AKKU 2 > AKKU 1 → VKE = "1". | 2120 |
| | > = F | | | N | J | N | x | x | | Vergleich zweier Festpunktzahlen auf größer oder gleich:<br>AKKU 2 ≥ AKKU 1 → VKE = "1". | 21A0 |
| | < F | | | N | J | N | x | x | | Vergleich zweier Festpunktzahlen auf kleiner:<br>AKKU 2 < AKKU 1 → VKE = "1". | 2140 |
| | < = F | | | N | J | N | x | x | | Vergleich zweier Festpunktzahlen auf kleiner oder gleich:<br>AKKU 2 ≤ AKKU 1 → VKE = "1". | 21C0 |
| | | | | | | | | | | **Verknüpfungs-Operationen (digital)** | |
| x | UW | | | N | N | N | x | x | | UND-Verknüpfung (wortweise): AKKU 2 and AKKU 1 = AKKU 1 | 4100 |
| x | OW | | | N | N | N | x | x | | ODER-Verknüpfung (wortweise): AKKU 2 or AKKU 1 AKKU 1 | 4900 |
| x | XOW | | | N | N | N | x | x | | Exclusiv-ODER-Verknüpfung (wortweise):<br>AKKU 2 mit AKKU 1 = AKKU 1 | 5100 |
| | | | | | | | | | | **Verknüpfungs-Operationen (binär)** | |
| | U | E | 0.0 ... 127.7 | N | J | N | | | | UND-Verknüpfung eines Eingangs auf Signalzustand "1" | $C0_i0_y0_y$ |
| | | A | 0.0 ... 127.7 | N | J | N | | | | UND-Verknüpfung eines Ausgangs auf Signalzustand "1" | $C0_i8_y0_y$ |
| | | M | 0.0 ... 255.7 | N | J | N | | | | UND-Verknüpfung eines Merkers auf Signalzustand "1" | $80_i0_y0_a$ |
| | | S | 0.0 ... 1023.7 | N | J | N | | | | UND-Verknüpfung eines S-Merkers auf Signalzustand "1" | 780B |
| | | T | 0 ... 127 | N | J | N | | | | UND-Verknüpfung eines Timers auf Signalzustand "1" | $F80_t0_t$ |
| | | Z | 0 ... 127 | N | J | N | | | | UND-Verknüpfung eines Zählers auf Signalzustand "1" | $B80_z0_z$ |
| | UN | E | 0.0 ... 127.7 | N | J | N | | | | UND-Verknüpfung eines Eingangs auf Signalzustand "0" | $E0_i0_y0_y$ |
| | | A | 0.0 ... 127.7 | N | J | N | | | | UND-Verknüpfung eines Ausgangs auf Signalzustand "0" | $E0_i8_y0_y$ |
| | | M | 0.0 ... 255.7 | N | J | N | | | | UND-Verknüpfung eines Merkers auf Signalzustand "0" | $A0_i0_y0_y$ |
| | | S | 0.0 ... 1023.7 | N | J | N | | | | UND-Verknüpfung eines S-Merkers auf Signalzustand "0" | 784B |
| | | T | 0 ... 127 | N | J | N | | | | UND-Verknüpfung eines Timers auf Signalzustand "0" | $FC0_t0_t$ |
| | | Z | 0 ... 127 | N | J | N | | | | UND-Verknüpfung eines Zählers auf Signalzustand "0" | $BC0_z0_z$ |
| | O | E | 0.0 ... 127.7 | N | J | N | | | | ODER-Verknüpfung eines Eingangs auf Signalzustand "1" | $C8_i0_y0_y$ |
| | | A | 0.0 ... 127.7 | N | J | N | | | | ODER-Verknüpfung eines Ausgangs auf Signalzustand "1" | $C8_i8_y0_y$ |
| | | M | 0.0 ... 255.7 | N | J | N | | | | ODER-Verknüpfung eines Merkers auf Signalzustand "1" | $88_i0_y0_y$ |
| | | S | 0.0 ... 1023.7 | N | J | N | | | | ODER-Verknüpfung eines S-Merkers auf Signalzustand "1" | 781B |
| | | T | 0 ... 127 | N | J | N | | | | ODER-Verknüpfung eines Timers auf Signalzustand "1" | $F90_t0_t$ |
| | | Z | 0 ... 127 | N | J | N | | | | ODER-Verknüpfung eines Zählers auf Signalzustand "1" | $B90_z0_z$ |
| | ON | E | 0.0 ... 127.7 | N | J | N | | | | ODER-Verknüpfung eines Eingangs auf Signalzustand "0" | $E8_i0_y0_y$ |
| | | A | 0.0 ... 127.7 | N | J | N | | | | ODER-Verknüpfung eines Ausgangs auf Signalzustand "0" | $E8_i8_y0_y$ |
| | | M | 0.0 ... 255.7 | N | J | N | | | | ODER-Verknüpfung eines Merkers auf Signalzustand "0" | $A8_i0_y0_y$ |
| | | S | 0.0 ... 1023.7 | N | J | N | | | | ODER-Verknüpfung eines S-Merkers auf Signalzustand "0" | 785B |
| | | T | 0 ... 127 | N | J | N | | | | ODER-Verknüpfung eines Timers auf Signalzustand "0" | $FD0_t0_t$ |
| | | Z | 0 ... 127 | N | J | N | | | | ODER-Verknüpfung eines Zählers auf Signalzustand "0" | $BD0_z0_z$ |
| | O | | | N | J | N | | | | ODER-Verknüpfung von UND-Funktionen. | FB00 |
| | U( | | | N | J | N | | | | UND-Verkn. von Klammerausdrücken (6 Klammerebenen) | BA00 |
| | O( | | | N | J | N | | | | ODER-Verkn. von Klammerausdrücken (6 Klammerebenen) | BB00 |
| | ) | | | N | J | N | | | | Klammer zu (Abschluss eines Klammerausdrucks) | BF00 |

| nur FB | Befehl | Operand | Parameter | VKE 1 abh. 2 beeinfl 3 begr. | | | Anz. | | | Funktionsbeschreibung | Befehls-Code (hex) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 | 1 | 0 | OV | | |
| x | U | =FOOP | | N | J | N | | | | UND-Verknüpfung eines Formaloperanden auf Signalzustand "1" (Parametertyp: BI, Aktualoperanden: E, A, M, T, Z) | $070_p0_p$ |
| x | UN | =FOOP | | N | J | N | | | | UND-Verknüpfung eines Formaloperanden auf Signalzustand "0" (Parametertyp: BI, Aktualoperanden: E, A, M, T, Z) | $270_p0_p$ |
| x | O | =FOOP | | N | J | N | | | | ODER-Verknüpfung: Formaloperanden auf Signalzustand "1" (Parametertyp: BI, Aktualoperanden: E, A, M, T, Z) | $0F0_p0_p$ |
| x | ON | =FOOP | | N | J | N | | | | ODER-Verknüpfung eines Formaloperanden auf Signalzustand "0" (Parametertyp: BI, Aktualoperanden: E, A, M, T, Z) | $2F0_p0_p$ |
| **Zähl-Operationen** | | | | | | | | | | | |
| | ZV | Z | 0 ... 127 | J | N | J | | | | Zähler zählt um 1 vorwärts. | $6C0_z0_z$ |
| | ZR | Z | 0 ... 127 | J | N | J | | | | Zähler zählt um 1 rückwärts. | $540_z0_z$ |
| | S | Z | 0 ... 127 | J | N | J | | | | Einen Zähler setzen. | $5C0_z0_z$ |
| | R | Z | 0 ... 127 | J | N | J | | | | Einen Zähler rücksetzen. | $7C0_z0_z$ |
| x | FR | Z | 0 ... 127 | J | N | J | | | | Zähler für Neustart freigeben. VKE=1 → Neustart Zähler | $440_z0_z$ |
| x | FR | =FOOP | | J | N | J | | | | Formaloperand (Zähler) für den Neustart freigeben. (Aktualoperanden: Z) | $060_p0_p$ |
| x | SVZ | =FOOP | | J | N | J | | | | Einen Zähler (Formaloperanden) setzen mit dem nachfolgenden angegebenen Zählwert. (Aktualoperand: Z) | $1E0_p0_p$ |
| x | SSV | =FOOP | | J | N | J | | | | Vorwärtszähler eines Zählers (Formaloperand). (Aktualoperand: Z) | $2E0_p0_p$ |
| x | SAR | =FOOP | | J | N | J | | | | Rückwärtszähler eines Zählers (Formaloperand). (Aktualop.: Z) | $160_p0_p$ |
| **Zeit-Operationen** | | | | | | | | | | | |
| | SI | T | 0 ... 127 | J | N | J | | | | Eine in AKKU 1 hinterlegte Zeit als verlängerten Impuls starten (Signalbegrenzung und -verlängerung) | $340_t0_t$ |
| | SV | T | 0 ... 127 | J | N | J | | | | Eine in AKKU 1 hinterlegte Zeit einschaltverzögernd starten | $1C0_t0_t$ |
| | SE | T | 0 ... 127 | J | N | J | | | | Eine in AKKU 1 hinterlegte Zeit als Impuls starten (Signalbegrenzung) | $240_t0_t$ |
| | SS | T | 0 ... 127 | J | N | J | | | | Eine in AKKU 1 hinterlegte Zeit speichernd einschaltverzögernd starten | $2C0_o0_t$ |
| | SA | T | 0 ... 127 | J | N | J | | | | Eine in AKKU 1 hinterlegte Zeit ausschaltverzögert starten | $140_t0_t$ |
| | R | T | 0 ... 127 | J | N | J | | | | Eine Zeit rücksetzen. | $3C0_t0_t$ |
| x | FR | T | 0 ... 127 | J | N | J | | | | Zeit für Neustart freigeben. VKE=1 → Neustart Zeit | $040_t0_t$ |
| x | FR | =FOOP | | J | N | J | | | | Formaloperand (Zeit) für den Neustart freigeben. (Aktualoperanden: T) | $060_p0_p$ |
| x | SI | =FOOP | | J | N | J | | | | Zeit (Formaloperand) als Impuls starten. Wert ist im AKKU 1 hinterlegt. (Aktualoperand: T) | $360_p0_p$ |
| x | SE | =FOOP | | J | N | J | | | | Zeit (Formaloperand) einschaltverzögernd starten. Wert ist im AKKU 1 hinterlegt. (Aktualoperand: T) | $260_p0_p$ |
| x | SVZ | =FOOP | | J | N | J | | | | Zeit (Formaloperand) als verlängerten Impuls starten mit dem im AKKU 1 hinterlegten Wert. (Aktualoperand: T) | $1E0_p0_p$ |
| x | SSV | =FOOP | | J | N | J | | | | Zeit (Formaloperand) als speichernde Einschaltverzögerung starten mit dem im AKKU 1 hinterlegten Wert. (Aktualop.: T) | $2E0_p0_p$ |
| x | SAR | =FOOP | | J | N | J | | | | Zeit (Formaloperand) als Ausschaltverzögerung starten mit dem im AKKU 1 hinterlegten Wert. (Aktualoperand: T) | $160_p0_p$ |

| nur FB | Befehl | Operand | Parameter | VKE 1 abh. 2 beeinfl 3 begr. | | | Anz. | | | Funktionsbeschreibung | Befehls-Code (hex) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 | 1 | 0 | OV | | |
| **Sonstige Operationen** | | | | | | | | | | | |
| x | AS | | | N | N | N | | | | Alarm sperren: Peripheriealarme und Zeit-OB-Bearbeitung wird gesperrt. | 0800 |
| x | AF | | | N | N | N | | | | Alarm freigeben: Hebt die Wirkung von AS wieder auf. | 0880 |
| x | B | DW | 0 ... 255 | N | N | N | | | | Datenwort bearbeiten: die nachfolgende Operation wird mit dem im Datenwort angegebenen Parameter kombiniert (ODER-Verknüpfung) und ausgeführt. | $6E0_o0_o$ |
| x | | MW | 0 ... 254 | N | N | N | | | | Merkerwort bearbeiten: die nachfolgende Operation wird mit dem im Merkerwort angegebenen Parameter kombiniert (ODER-Verknüpfung) und ausgeführt. | $4E0_o0_o$ |
| x | BI | =FOOP | | N | N | N | | | | Über einen Formaloperanden bearbeiten (indirekt). Die Nummer des Formaloperanden steht im AKKU 1. | 7E00 |
| x | TAK | | | N | N | N | | | | Den Inhalt von AKKU 1 und AKKU 2 tauschen. | 7002 |

Erläuterungen zu den Indizes des Befehlscodes

| | |
|---|---|
| y | + Byteadresse |
| i | + Bitadresse |
| p | + Parameteradresse |
| t | + Zeitgliednummer |
| k | + Konstante |
| n | + Bausteinnummer |
| o | + Wortadresse |
| s | + Schiebezahl |
| m | + relative Sprungadresse |
| r | + Registeradresse |
| l | + Blocklänge in Byte |
| j | + Sprungweite (16 Bit) |
| w | + Wert |
| z | + Zählernummer |

# B Maschinencode

| Maschinen-Code | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| B0 | | B1 | | B2 | | B3 | | Operation | Operand |
| L | R | L | R | L | R | L | R | | |
| 0 | 0 | 0 | 0 | | | | | NOP 0 | |
| 0 | 1 | 0 | 0 | | | | | KEW | |
| 0 | 2 | $0_d$ | $0_d$ | | | | | L | T |
| 0 | 3 | $0_l$ | $0_l$ | | | | | TNB | |
| 0 | 4 | $0_t$ | $0_t$ | | | | | FR | T |
| 0 | 5 | 0 | 0 | | | | | BEB | |
| 0 | 6 | $0_p$ | $0_p$ | | | | | FR = | |
| 0 | 7 | $0_p$ | $0_p$ | | | | | U = | |
| 0 | 8 | 0 | 0 | | | | | AS | |
| 0 | 8 | 8 | 0 | | | | | AF | |
| 0 | 9 | 0 | 0 | | | | | KZW | |
| 0 | A | $0_y$ | $0_y$ | | | | | L | MB |
| 0 | B | $0_y$ | $0_y$ | | | | | T | MB |
| 0 | C | $0_t$ | $0_t$ | | | | | LC | T |
| 0 | D | $0_m$ | $0_m$ | | | | | SPO = | |
| 0 | E | $0_p$ | $0_p$ | | | | | LC = | |
| 0 | F | $0_p$ | $0_p$ | | | | | O = | |
| 1 | 0 | 8 | 2 | | | | | BLD | 130 |
| 1 | 0 | 8 | 3 | | | | | BLD | 131 |
| 1 | 0 | 8 | 4 | | | | | BLD | 132 |
| 1 | 0 | 8 | 5 | | | | | BLD | 133 |
| 1 | 0 | F | F | | | | | BLD | 255 |
| 1 | 1 | $0_w$ | $0_w$ | | | | | I | |
| 1 | 2 | $0_y$ | $0_y$ | | | | | L | MW |
| 1 | 3 | $0_y$ | $0_y$ | | | | | T | MW |
| 1 | 4 | $0_t$ | $0_t$ | | | | | SA | T |
| 1 | 5 | $0_m$ | $0_m$ | | | | | SPP = | |
| 1 | 6 | $0_p$ | $0_p$ | | | | | SAR = | |
| 1 | 7 | $0_p$ | $0_p$ | | | | | S = | |
| 1 | 9 | $0_w$ | $0_w$ | | | | | D | |
| 1 | C | $0_t$ | $0_t$ | | | | | SV | T |
| 1 | D | $0_n$ | $0_n$ | | | | | SPB | FB |
| 1 | E | $0_p$ | $0_p$ | | | | | SVZ = | |
| 1 | F | $0_p$ | $0_p$ | | | | | = = | |
| 2 | 0 | $0_n$ | $0_n$ | | | | | A | DB |
| 2 | 1 | 2 | 0 | | | | | > F | |
| 2 | 1 | 4 | 0 | | | | | < F | |
| 2 | 1 | 6 | 0 | | | | | >< F | |
| 2 | 1 | 8 | 0 | | | | | ! = F | |
| 2 | 1 | A | 0 | | | | | > = F | |
| 2 | 1 | C | 0 | | | | | < = F | |
| 2 | 2 | $0_o$ | $0_o$ | | | | | L | DL |
| 2 | 3 | $0_o$ | $0_o$ | | | | | T | DL |
| 2 | 4 | $0_t$ | $0_t$ | | | | | SE | T |
| 2 | 5 | $0_m$ | $0_m$ | | | | | SPM = | |
| 2 | 6 | $0_p$ | $0_p$ | | | | | SE = | |
| 2 | 7 | $0_p$ | $0_p$ | | | | | UN = | |
| 2 | 8 | $0_k$ | $0_k$ | | | | | L | KB |
| 2 | A | $0_o$ | $0_o$ | | | | | L | DR |
| 2 | B | $0_o$ | $0_o$ | | | | | T | DR |
| 2 | C | $0_o$ | $0_t$ | | | | | SS | T |
| 2 | D | $0_m$ | $0_m$ | | | | | SPA = | |
| 2 | E | $0_p$ | $0_p$ | | | | | SSV = | |
| 2 | F | $0_p$ | $0_p$ | | | | | ON = | |

| Maschinen-Code | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| B0 | | B1 | | B2 | | B3 | | Operation | Operand |
| L | R | L | R | L | R | L | R | | |
| 3 | 0 | 0 | 1 | $0_k$ | $0_k$ | $0_k$ | $0_k$ | L | KZ |
| 3 | 0 | 0 | 2 | $0_k$ | $0_k$ | $0_k$ | $0_k$ | L | KT |
| 3 | 0 | 0 | 4 | $0_k$ | $0_k$ | $0_k$ | $0_k$ | L | KF |
| 3 | 0 | 1 | 0 | $0_k$ | $0_k$ | $0_k$ | $0_k$ | L | KC |
| 3 | 0 | 2 | 0 | $0_k$ | $0_k$ | $0_k$ | $0_k$ | L | KY |
| 3 | 0 | 4 | 0 | $0_k$ | $0_k$ | $0_k$ | $0_k$ | L | KH |
| 3 | 0 | 8 | 0 | $0_k$ | $0_k$ | $0_k$ | $0_k$ | L | KM |
| 3 | 2 | $0_o$ | $0_o$ | | | | | L | DW |
| 3 | 3 | $0_o$ | $0_o$ | | | | | T | DW |
| 3 | 4 | $0_t$ | $0_t$ | | | | | SI | T |
| 3 | 5 | $0_m$ | $0_m$ | | | | | SPN = | |
| 3 | 6 | $0_p$ | $0_p$ | | | | | SI = | |
| 3 | 7 | $0_p$ | $0_p$ | | | | | RB = | |
| 3 | C | $0_t$ | $0_t$ | | | | | R | T |
| 3 | D | $0_n$ | $0_n$ | | | | | SPA | FB |
| 3 | E | $0_p$ | $0_p$ | | | | | RD = | |
| 3 | F | $0_p$ | $0_p$ | | | | | LW = | |
| 4 | 0 | 0 | $0_r$ | | | | | LIR | |
| 4 | 1 | 0 | 0 | | | | | UW | |
| 4 | 2 | $0_z$ | $0_z$ | | | | | L | Z |
| 4 | 4 | $0_z$ | $0_z$ | | | | | FR | Z |
| 4 | 5 | $0_m$ | $0_m$ | | | | | SPZ = | |
| 4 | 6 | $0_p$ | $0_p$ | | | | | L = | |
| 4 | 8 | 0 | $0_r$ | | | | | TIR | |
| 4 | 9 | 0 | 0 | | | | | OW | |
| 4 | A | $0_y$ | $0_y$ | | | | | L | EB |
| 4 | A | $8_y$ | $0_y$ | | | | | L | AB |
| 4 | B | $0_y$ | $0_y$ | | | | | T | EB |
| 4 | B | $8_y$ | $0_y$ | | | | | T | AB |
| 4 | C | $0_z$ | $0_z$ | | | | | LC | Z |
| 4 | D | $0_n$ | $0_n$ | | | | | SPB | OB |
| 4 | E | $0_o$ | $0_o$ | | | | | B | MW |
| 5 | 0 | $0_k$ | $0_k$ | | | | | ADD | BF |
| 5 | 1 | 0 | 0 | | | | | XOW | |
| 5 | 2 | $0_y$ | $0_y$ | | | | | L | EW |
| 5 | 2 | $8_y$ | $0_y$ | | | | | L | AW |
| 5 | 3 | $0_y$ | $0_y$ | | | | | T | EW |
| 5 | 3 | $8_y$ | $0_y$ | | | | | T | AW |
| 5 | 4 | $0_z$ | $0_z$ | | | | | ZR | Z |
| 5 | 5 | $0_n$ | $0_n$ | | | | | SPB | PB |
| 5 | 8 | 0 | 0 | $0_k$ | $0_k$ | $0_k$ | $0_k$ | ADD | KF |
| 5 | 9 | 0 | 0 | | | | | -F | |
| 5 | C | $0_z$ | $0_z$ | | | | | S | Z |
| 5 | D | $0_n$ | $0_n$ | | | | | SPB | SB |
| 6 | 1 | $0_s$ | $0_s$ | | | | | SLW | |
| 6 | 2 | $0_o$ | $0_o$ | | | | | L | BS |
| 6 | 3 | $0_o$ | $0_o$ | | | | | T | BS |
| 6 | 5 | 0 | 0 | | | | | BE | |
| 6 | 5 | 0 | 1 | | | | | BEA | |
| 6 | 6 | $0_p$ | $0_p$ | | | | | T = | |
| 6 | 8 | 0 | B | | | | | LDI | A1 |
| 6 | 8 | 0 | F | | | | | TDI | A1 |
| 6 | 8 | 2 | B | | | | | LDI | A2 |
| 6 | 8 | 2 | F | | | | | TDI | A2 |
| 6 | 9 | $0_s$ | $0_s$ | | | | | SRW | |
| 6 | C | $0_z$ | $0_z$ | | | | | ZV | Z |
| 6 | D | $0_n$ | $0_n$ | | | | | SPA | OB |
| 6 | E | $0_o$ | $0_o$ | | | | | B | DW |

| Maschinen-Code | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| B0 | | B1 | | B2 | | B3 | | Operation | Operand |
| L | R | L | R | L | R | L | R | | |
| 7 | 0 | 0 | 0 | | | | | STS | |
| 7 | 0 | 0 | 2 | | | | | TAK | |
| 7 | 0 | 0 | 3 | | | | | STP | |
| 7 | 0 | 0 | B | $0_j$ | $0_j$ | $0_j$ | $0_j$ | SPR | |
| 7 | 0 | 1 | 5 | C | 0 | $0_z$ | $0_z$ | P | Z |
| 7 | 0 | 1 | 5 | 8 | 0 | $0_z$ | $0_z$ | PN | Z |
| 7 | 0 | 1 | 5 | 4 | 0 | $0_z$ | $0_z$ | SU | Z |
| 7 | 0 | 1 | 5 | 0 | 0 | $0_z$ | $0_z$ | RU | Z |
| 7 | 0 | 2 | 5 | C | 0 | $0_t$ | $0_t$ | P | T |
| 7 | 0 | 2 | 5 | 8 | 0 | $0_t$ | $0_t$ | PN | T |
| 7 | 0 | 2 | 5 | 4 | 0 | $0_t$ | $0_t$ | SU | T |
| 7 | 0 | 2 | 5 | 0 | 0 | $0_t$ | $0_t$ | RU | T |
| 7 | 0 | 4 | 6 | C | $0_i$ | $0_o$ | $0_o$ | P | D |
| 7 | 0 | 4 | 6 | 8 | $0_i$ | $0_o$ | $0_o$ | PN | D |
| 7 | 0 | 4 | 6 | 4 | $0_i$ | $0_o$ | $0_o$ | SU | D |
| 7 | 0 | 4 | 6 | 0 | $0_i$ | $0_o$ | $0_o$ | RU | D |
| 7 | 0 | 5 | 7 | C | $0_i$ | $0_o$ | $0_o$ | P | BS |
| 7 | 0 | 5 | 7 | 8 | $0_i$ | $0_o$ | $0_o$ | PN | BS |
| 7 | 0 | 5 | 7 | 4 | $0_i$ | $0_o$ | $0_o$ | SU | BS |
| 7 | 0 | 5 | 7 | 0 | $0_i$ | $0_o$ | $0_o$ | RU | BS |
| 7 | 2 | $0_y$ | $0_y$ | | | | | L | PB/PY* |
| 7 | 3 | $0_y$ | $0_y$ | | | | | T | PB/PY* |
| 7 | 5 | $0_n$ | $0_n$ | | | | | SPA | PB |
| 7 | 6 | $0_p$ | $0_p$ | | | | | | B = |
| 7 | 8 | 0 | 5 | 0 | 0 | $0_n$ | $0_n$ | E | DB |
| 7 | 8 | 0 | B | | | | | U | S |
| 7 | 8 | 1 | B | | | | | O | S |
| 7 | 8 | 2 | B | $0_i$ | $0_y$ | $0_y$ | $0_y$ | S | S |
| 7 | 8 | 3 | B | $0_i$ | $0_y$ | $0_y$ | $0_y$ | = | S |
| 7 | 8 | 4 | B | | | | | UN | S |
| 7 | 8 | 5 | B | | | | | ON | S |
| 7 | 8 | 6 | B | $0_i$ | $0_y$ | $0_y$ | $0_y$ | R | S |
| 7 | 8 | A | B | 0 | $0_y$ | $0_y$ | $0_y$ | L | SY |
| 7 | 8 | C | B | 0 | $0_y$ | $0_y$ | $0_y$ | L | SW |
| 7 | 9 | 0 | 0 | | | | | + F | |
| 7 | A | $0_y$ | $0_y$ | | | | | L | PW |
| 7 | B | $0_y$ | $0_y$ | | | | | T | PW |
| 7 | C | $0_z$ | $0_z$ | | | | | R | Z |
| 7 | D | $0_n$ | $0_n$ | | | | | SPA | SB |
| 7 | E | 0 | 0 | | | | | BI | |
| 8 | $0_i$ | $0_y$ | $0_a$ | | | | | U | M |
| 8 | $8_i$ | $0_y$ | $0_y$ | | | | | O | M |
| 9 | $0_i$ | $0_y$ | $0_y$ | | | | | S | M |
| 9 | $8_i$ | $0_y$ | $0_y$ | | | | | = | M |
| A | $0_i$ | $0_y$ | $0_y$ | | | | | UN | M |
| A | $8_i$ | $0_y$ | $0_y$ | | | | | ON | M |
| B | $0_i$ | $0_y$ | $0_y$ | | | | | R | M |
| B | 8 | $0_z$ | $0_z$ | | | | | U | Z |
| B | 9 | $0_z$ | $0_z$ | | | | | O | Z |
| B | A | 0 | 0 | | | | | U( | |
| B | B | 0 | 0 | | | | | O( | |
| B | C | $0_z$ | $0_z$ | | | | | UN | Z |
| B | D | $0_z$ | $0_z$ | | | | | ON | Z |
| B | F | 0 | 0 | | | | | ) | |
| C | $0_i$ | $0_y$ | $0_y$ | | | | | U | E |
| C | $0_i$ | $8_y$ | $0_y$ | | | | | U | A |
| C | $8_i$ | $0_y$ | $0_y$ | | | | | O | E |
| C | $8_i$ | $8_y$ | $0_y$ | | | | | O | A |

| Maschinen-Code | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| B0 | | B1 | | B2 | | B3 | | Operation | Operand |
| L | R | L | R | L | R | L | R | | |
| D | $0_i$ | $0_y$ | $0_y$ | | | | | S | E |
| D | $0_i$ | $8_y$ | $0_y$ | | | | | S | A |
| D | $8_i$ | $0_y$ | $0_y$ | | | | | = | E |
| D | $8_i$ | $8_y$ | $0_y$ | | | | | = | A |
| E | $0_i$ | $0_y$ | $0_y$ | | | | | UN | E |
| E | $0_i$ | $8_y$ | $0_y$ | | | | | UN | A |
| E | $8_i$ | $0_y$ | $0_y$ | | | | | ON | E |
| E | $8_i$ | $8_y$ | $0_y$ | | | | | ON | A |
| F | $0_i$ | $0_y$ | $0_y$ | | | | | R | E |
| F | $0_i$ | $8_y$ | $0_y$ | | | | | R | A |
| F | 8 | $0_t$ | $0_t$ | | | | | U | T |
| F | 9 | $0_t$ | $0_t$ | | | | | O | T |
| F | A | $0_m$ | $0_m$ | | | | | SPB = | |
| F | B | 0 | 0 | | | | | O | |
| F | C | $0_t$ | $0_t$ | | | | | UN | T |
| F | D | $0_t$ | $0_t$ | | | | | ON | T |
| F | F | F | F | | | | | NOP 1 | |

Erläuterungen zu den Indizes des Befehlscodes

y + Byteadresse
i + Bitadresse
p + Parameteradresse
t + Zeitgliednummer
k + Konstante
n + Bausteinnummer
o + Wortadresse
s + Schiebezahl
m + relative Sprungadresse
r + Registeradresse
l + Blocklänge in Byte
j + Sprungweite (16 bit)
w + Wert
z + Zählernummer

# C Index


***1***

15pol PG/OP-SubD-Buchse ..... 2-10

***2***

20mA-Schnittstelle ..... 2-11

***A***

Adressierung über DB1 ..... 3-5
Alarm ..... 12-1
  Baugruppen ..... 12-2
  Bearbeitung ..... 7-14, 8-27
  Diagnosedaten ..... 12-11
  Interrupt ..... 12-8
  Priorität und Reaktionszeit ..... 12-7
  Übersicht ..... 12-2
  Unterbrechung ..... 12-2
  Weckalarm ..... 12-9
Analogwertverarbeitung ..... 10-9
  Beispiel ..... 10-14
Anlauf ..... 7-8
  Anwenderschnittstellen ..... 7-10
  Unterbrechung ..... 7-10
A-NR ..... 10-19
Anwendungsschicht ..... 4-5
ANZ 0 ..... 9-76
ANZ 1 ..... 9-76
Anzeigenbildung ANZ, OV ..... 9-76
ANZW ..... 10-19
ANZW Anzeigenwort ..... 5-36, 10-27
  Zustände ..... 10-34
Aufbau der Bausteine ..... 8-12
AWL Anweisungsliste ..... 8-4

***B***

Bausteine ..... 8-9
  Arten ..... 8-9
  bearbeiten ..... 8-32
  Hantierungsbausteine ..... 10-17
  integrierte Bausteine ..... 10-1
Betriebsartenschalter ..... 2-8
Betriebsstundenzähler ..... 11-22
Betriebszustände ..... 2-8, 7-1, 7-2
  Anlauf ..... 7-8
  RUN ..... 7-11
  STOP ..... 7-6
Bitübertragungsschicht ..... 4-4
BLGR ..... 10-20
Blockgröße ..... 5-15, 10-37

***C***

CPU 24x
  Anlaufverhalten ..... 2-18
  Arbeitsweisen ..... 1-7
  Aufbau ..... 2-8
  Blockschaltbild ..... 2-16
  Eigenschaften ..... 2-5
  FBs, OBs, Sonderbefehle ..... 2-34
  LEDs ..... 2-8
  Operanden ..... 1-8
  Pinbelegung ..... 2-10
  Programm internen sichern ..... 2-20
  Spannungsversorgung ..... 2-9
  Speicherbereiche ..... 2-25
  Technische Daten ..... 2-36
CPU 24x DP ..... 2-4
  Adresseinsteller ..... 2-15
  Adressierung ..... 6-9
  Aufbaurichtlinien ..... 6-19
  Beispiel ..... 6-28
  Beispielnetze ..... 6-22
  Busabschluss ..... 6-26
  Datenkonsistenz ..... 6-5
  Datenverkehr ..... 6-3
  Diagnose ..... 6-14
    gerätebezogen ..... 6-16
    LEDs ..... 6-27
    Norm- ..... 6-15
  GSD Gerätestammdatei ..... 6-8
  Ident-Nummer ..... 6-8
  Inbetriebnahme ..... 6-24
  Initialisierungsphase ..... 6-26
  LEDs ..... 2-14
  Leitungsabschluss ..... 6-24
  Parameterdaten ..... 6-12
  Projektierung ..... 6-6
  Statusmeldung ..... 6-17
  Technische Daten ..... 2-38
  Token-Passing-Verfahren ..... 6-3
  Übertragungsprotokoll ..... 6-3
  Verkabelung ..... 6-25
CPU 24x-2BT01 ..... 2-3
  Adresse
    Broadcast- ..... 5-10
    Erstinbetriebnahme ..... 5-11
    Ethernet/IP- ..... 5-10
  Anlauf ..... 5-29
  Anwendung ..... 5-6
  ANZW Anzeigenwort ..... 5-36
  Ethernet interface ..... 2-13
  Fehleranzeige ..... 5-36
  Jokerlänge ..... 5-35
  Kommunikation ..... 5-3
  Kopplung ..... 5-32
  LEDs ..... 2-12
  NETZEIN ..... 5-15
  Netzwerkplanung ..... 5-7
  ORG-Format ..... 5-32
  PAFE Param.fehler ..... 5-43
  Projektierung ..... 5-12
    Beispiel ..... 5-16
  SPS-Header ..... 5-34
  SPS-Programmierung ..... 5-14
  Synchronisation ..... 5-15
  Systemeigenschaften ..... 5-30
  Technische Daten ..... 2-37
  TRADA ..... 5-35
CPU 24x-2BT10 ..... 2-3

Adresse
Erstinbetriebnahme ..... 4-10
IP 4-10
IP-Klassen ..... 4-11
Ethernet interface ..... 2-13
Fehler
Diagnose ..... 4-30
Meldungen ..... 4-26
Fremdsysteme ..... 4-27
Funktionsübersicht ..... 4-15
Kommunikation ..... 4-14
Programmierung ..... 4-16
Typen ..... 4-14
Kopplung ..... 4-27
LEDs ..... 2-12
NCM Diagnose ..... 4-30
ORG-Format ..... 4-27
Protokolle ..... 4-7
RFC1006 ..... 4-9
SPS
Anwenderprogramm ..... 4-23
Header ..... 4-29
Subnet-Mask ..... 4-10
Technische Daten ..... 2-37

***D***

Darstellungsarten
AWL, KOP, FUP ..... 8-4
Darstellungsschicht ..... 4-5
Datenbausteine ..... 8-20
DB 1 ..... 10-61
DBNR ..... 10-19

***E***

Einsatz von E/A-Modulen ..... 3-2
Einsatzbereiche ..... 1-4
Ergänzende Operationen ..... 9-40
ERW-Kennung ..... 4-27
Ethernet
Adresse ..... 5-10
Normen ..... 5-9

***F***

Fehler
CPU 24x
Diagnose im Programm ..... 8-35
Reaktions-OBs ..... 8-29
Statuswort ..... 11-10
USTACK ..... 2-27
Weckfehler ..... 12-10
CPU 24x DP
Diagnose ..... 6-14
CPU 24x-2BT01
ANZW Anzeigenwort ..... 5-36
PAFE Param.fehler ..... 5-43
Funktionsbaustein ..... 8-15
FB 238 COMPR ..... 10-3
FB 239 DELETE ..... 10-4
FB 241 COD.16 ..... 10-6
FB 242 Multiplizierer ..... 10-7
FB 243 Dividierer ..... 10-8
FB 244 SEND ..... 10-38
FB 245 RECEIVE ..... 10-41
FB 246 FETCH ..... 10-44
FB 247 CONTROL ..... 10-45
FB 248 RESET ..... 10-46
FB 249 SYNCHRON ..... 10-47
FB 250 RLG AE ..... 10-9
FB 251 RLG AA ..... 10-12
FUP Funktionsplan ..... 8-5

***G***

Gerätevarianten ..... 1-6
Grundlagen ..... 1-1
Grundoperationen ..... 9-3
GSD-Datei ..... 6-8

***H***

H1 ..... 5-4
Hantierungsbausteine ..... 10-17
Blockgröße ..... 10-37
FBs ..... 10-38
Fehleranzeige ANZW ..... 10-27
Längen-Wort ..... 10-35
Parameter ..... 10-18
indirekte ..... 10-24
QTYP/ZTYP ..... 10-25
Hardwarebeschreibung ..... 2-1
Host-ID ..... 4-11
Hub ..... 4-6, 5-2

***I***

Ident-Nummer ..... 6-8
Inbetriebnahme ..... 2-17
Industrial Ethernet ..... 5-4
Integrierte
Datenbaustein DB 1 ..... 10-61
FBs ..... 10-2
Sonder-OBs ..... 10-48
IP-Adresse ..... 5-10
ISO/OSI-Schichtenmodell ..... 4-3

***J***

Jokerlänge ..... 5-35

***K***

Kommunikationsebenen ..... 4-2
Komponenten ..... 2-8
Komprimieren ..... 8-33
KOP Kontaktplan ..... 8-5

***L***

Längen-Wort ..... 10-35
Leistungsmerkmale ..... 1-5

***N***

NCM-Diagnose ..... 4-30
Netzwerk ..... 4-6, 5-2
Schicht ..... 4-4
Neustart ..... 2-19, 7-9

***O***

ON/OFF-Schalter ..... 2-9
Operandenbereiche ..... 8-6

Operationsarten ..... 8-6
ORG Organisationsformat ..... 5-32
Organisationsbaustein ..... 8-13, 8-25
OB 1 ..... 7-12, 8-25
OB 10 bis 13 ..... 12-4
OB 160 ..... 10-50
OB 19 ..... 8-29
OB 21 ..... 7-10
OB 22 ..... 7-10
OB 23 ..... 8-29
OB 24 ..... 8-29
OB 251 ..... 10-51
OB 27 ..... 8-30
OB 31 ..... 10-49
OB 32 ..... 8-30
OB 33 ..... 8-30
OB 34 ..... 8-31
OB 6 ..... 8-27, 12-6
ORG-Format ..... 4-27
OV Overflow ..... 9-76

***P***

PAFE ..... 5-43, 10-20
Peripheriemodule
Adressierung ..... 3-3
Parametrierung ..... 3-6
Profibus
DP ..... 6-2
Profibus DP-Master
Abisolierlängen ..... 6-21
Leitungsabschluss ..... 6-21
Stecker ..... 6-20
Programmbaustein ..... 8-15
Programmbearbeitung ..... 8-22
Ebenen ..... 7-3
zeitdauergesteuert ..... 7-14, 12-6
zeitgesteuert ..... 7-15, 8-25, 12-4
zyklisch ..... 7-12, 8-25
Programmstruktur ..... 8-8

***Q***

QANF/ZANF ..... 10-20
QLAE/ZLAE ..... 10-20
QTYP/ZTYP ..... 10-19
QTYP/ZTYP - Parameter ..... 10-25

***R***

RECV1 (FB 235) ..... 4-25
RFC1006 ..... 4-9
RUN ..... 7-11

***S***

Schrittbaustein ..... 8-15
SEND1 (FB 234) ..... 4-25
Sicherungsschicht ..... 4-4
Sitzungsschicht ..... 4-5
Sonder-OBs ..... 10-48
Speicher
Komprimieren ..... 8-33
-Modul ..... 2-24
SPS Header ..... 4-29
SPS-Header ..... 5-34
SSNR ..... 10-19
STATUS ..... 2-22
Anzeige ..... 10-27
VAR ..... 2-23
Sternkoppler ..... 5-2
Sterntopologie ..... 2-13
Steuerbit-Abkürzungen ..... 2-29
STEUERN ..... 2-23
Steuern VAR ..... 5-27
VAR ..... 2-23
STOP ..... 7-6
Switch ..... 4-6
Synchronisation der Schnittst ..... 5-21
System ..... 2-2
Daten ..... 2-26
Operationen ..... 9-66
Übersicht ..... 2-2

***T***

TCP/IP ..... 4-7, 5-5
Testprogramm ..... 5-44
TRADA ..... 5-35
Transportschicht ..... 4-4
Twisted Pair ..... 4-6, 5-2
Einschränkungen ..... 5-8

***U***

UDP ..... 4-9
Uhr ..... 11-2
Betriebsstundenzähler ..... 11-22
Datenbereich. ..... 11-7
Funktionen ..... 11-1
Initialisierung ..... 11-4
Korrekturwert ..... 11-6
Parametrierung ..... 11-3
Pufferung ..... 11-2
Statuswort ..... 11-10
stellen/lesen ..... 11-12
Uhrzeit / Datum lesen ..... 11-12
Weckfunktionen ..... 11-19
Urlöschen ..... 7-7
URLÖSCHEN ..... 2-19
USTACK ..... 2-27

***V***

Voreinstellungs-Schalter ..... 2-8

***W***

Weckalarm ..... 12-9
Weckfehler ..... 7-16, 12-10
Weckfunktionen ..... 11-19
Wiederanlauf ..... 7-9

***Z***

Zahlendarstellung ..... 8-34
Zähloperationen ..... 9-23
Zeitoperationen ..... 9-13